open source PRESS

synopsis

Mesh

Drahtlose Ad-hoc-Netze

Corinna „Elektra" Aichele

open
source
PRESS

Corinna „Elektra“ Aichele

Mesh

Drahtlose Ad-hoc-Netze

Alle in diesem Buch enthaltenen Programme, Darstellungen und Informationen wurden nach bestem Wissen erstellt. Dennoch sind Fehler nicht ganz auszuschließen. Aus diesem Grunde sind die in dem vorliegenden Buch enthaltenen Informationen mit keiner Verpflichtung oder Garantie irgendeiner Art verbunden. Autor(en), Herausgeber, Übersetzer und Verlag übernehmen infolgedessen keine Verantwortung und werden keine daraus folgende Haftung übernehmen, die auf irgendeine Art aus der Benutzung dieser Informationen – oder Teilen davon – entsteht, auch nicht für die Verletzung von Patentrechten, die daraus resultieren können. Ebenso wenig übernehmen Autor(en) und Verlag die Gewähr dafür, dass die beschriebenen Verfahren usw. frei von Schutzrechten Dritter sind.
Die in diesem Werk wiedergegebenen Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. werden ohne Gewährleistung der freien Verwendbarkeit benutzt und können auch ohne besondere Kennzeichnung eingetragene Marken oder Warenzeichen sein und als solche den gesetzlichen Bestimmungen unterliegen.
Dieses Werk ist urheberrechtlich geschützt. Alle Rechte, auch die der Übersetzung, des Nachdrucks und der Vervielfältigung des Buches – oder Teilen daraus – vorbehalten. Kein Teil des Werkes darf ohne schriftliche Genehmigung des Verlags in irgendeiner Form (Druck, Fotokopie, Mikrofilm oder einem anderen Verfahren), auch nicht für Zwecke der Unterrichtsgestaltung, reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

Bibliografische Information Der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

© 2007 Open Source Press, München
Gesamtlektorat: Ulrich Wolf
Satz: Open Source Press (LaTeX)
Umschlaggestaltung: www.fritzdesign.de
Gesamtherstellung: Kösel, Krugzell

ISBN 978-3-937514-39-0

http://www.opensourcepress.de

Inhaltsverzeichnis

Vorwort 11

1 Einleitung 13

2 Meshrouting 19

2.1 Ad-Hoc vs. Infrastruktur 20

2.1.1 Bandbreitenproblematik bei Multihop-Links mit nur einem Interface 22

2.1.2 Problemfall TCP 24

2.1.3 Exposed Nodes – Hidden Nodes 26

2.2 Routingprotokolle für drahtlose Netzwerke 27

2.2.1 Protokolltypen 29

2.2.2 Praktische Relevanz der Routingprotokolle 31

3 Optimized Link State Routing 35

3.1 Entwicklungsgeschichte 36

3.2 Die OLSR-Implementierung von Olsr.org und Freifunk 37

3.2.1 Hysteresealgorithmus 37

3.2.2 Multipoint-Relais 38

3.2.3 Olsr.org – Aus den Fehlern lernen 39

3.2.4 Funktionsweise des OLSR-Daemon 39

3.2.5 Topologieinformationen und Dijkstra-Algorithmus . . . 40

3.3 Praxis mit Olsrd 44

3.3.1 Installation unter Linux, Mac OS X und *BSD 44

3.3.2 Installation unter Windows 46

3.3.3 Konfigurationsdatei des Olsrd 48

3.3.4 Erster Start des Daemon 57
3.4 Wichtige Plugins für Olsrd 61
3.4.1 DotDraw 62
3.4.2 Httpinfo 63
3.4.3 Nameservice 65
3.5 Typische Probleme mit OLSR 67
3.5.1 Gateway-Switching 67
3.5.2 Typische Probleme beim ersten Einsatz 69
3.5.3 Probleme mit WiFi 69

4 B.A.T.M.A.N. – Better Approach To Mobile Ad-Hoc Networking 71
4.1 Funktionsweise des B.A.T.M.A.N.-Algorithmus 73
4.1.1 Die Originatornachricht 73
4.1.2 Bidirectional Link Check – Linkprüfung auf Bidirektionalität 74
4.1.3 Eine Originatornachricht reist durch das Mesh 76
4.2 Praxis mit B.A.T.M.A.N. 79
4.2.1 Erster Start des Daemons 80
4.2.2 Den Netzwerkstatus im Debugging-Modus beobachten 81
4.2.3 Routingklassen und Gatewayklassen 83
4.2.4 Visualisierungsserver 85
4.2.5 Firewalleinstellungen anpassen 86
4.2.6 Webinterface für B.A.T.M.A.N. 86

5 WLAN-Technik 87
5.1 Die WiFi-Standards 802.11 von a bis Draft-n 88
5.1.1 IEEE 802.11b/g 88
5.1.2 IEEE 802.11a 90
5.1.3 IEEE 802.11n 92
5.1.4 Bruttodatenrate und Nettodatenrate 92
5.2 Geräte für den Aufbau eines Meshnetzwerks 93
5.2.1 SoHO-Router oder PC? 93
5.2.2 WLAN-Karten für PCs und Routerboards 94
5.2.3 SoHO-WLAN-Router 95
5.2.4 Professionelle WLAN-Router 96

5.2.5 WLAN-Router im Selbstbau 96
5.2.6 HF-Steckverbinder 97
5.2.7 Koaxialkabel 98
5.2.8 Gehäuse 99
5.2.9 Typische Hardware-Probleme im Ad-Hoc-Modus 101
5.3 Chipsätze 104
5.3.1 Prism für 802.11b 104
5.3.2 Lucent-Orinoco 105
5.3.3 Prism54 Hard- und Soft-MAC 106
5.3.4 Atheros 106
5.3.5 Atmel 108
5.3.6 Broadcom 108
5.3.7 Cisco 109
5.3.8 Intel 109
5.3.9 Ralink 110
5.3.10 Realtek 111
5.3.11 Texas Instruments 111
5.3.12 Zydas 111
5.4 Typische Hardware-Probleme im Ad-Hoc-Modus 112
5.4.1 Cell-Splitting 112
5.4.2 Der IBSSID-Hack 115
5.5 Power over Ethernet 116
5.5.1 Einfaches PoE im Selbstbau 116
5.5.2 Einfache PoE-Lösungen selbst entwickeln und berechnen 118
5.5.3 Spannungsverluste auf der Leitung berechnen 122

6 Mesh-Betrieb auf SoHO-Routern 125

6.1 Freifunk-kompatible Router 126
6.1.1 Linksys WRT54GL und WRT54G 128
6.1.2 Buffalo WHR-G54S 129
6.1.3 Asus WL500G Premium 130
6.2 Installation der Freifunk-Firmware 130
6.2.1 Firmware-Installation auf Linksys WRT54GL 131

6.2.2 Firmware-Installation per TFTP 134
6.3 Konfiguration per Webinterface 135
6.3.1 Sicher anmelden über SSH 136
6.3.2 Konfigurationsschritte 137
6.3.3 Kennwort und Kontaktinfos 137
6.3.4 System 138
6.3.5 OLSR 139
6.3.6 Drahtlos 143
6.3.7 LAN 148
6.3.8 WAN 150
6.3.9 Publizieren 151
6.3.10 Software1 151
6.3.11 Software2 152
6.3.12 Firmware 152
6.3.13 Neustart 152
6.4 Gateway-Konfiguration 153
6.4.1 Internet-Gateway über DSL 153
6.4.2 Internet-Gateway über ein bereits existierendes LAN . . 153
6.4.3 Fortgeschrittene Konfiguration – das Freifunk-Gateway-Plugin 154
6.5 Konfiguration auf der Kommandozeile 156
6.5.1 Softwareinstallation auf der Kommandozeile 156
6.5.2 Zugang vom Mesh ins LAN 157
6.6 OpenWRT 159
6.7 Wenn etwas schiefgeht – Probleme mit der Router-Firmware . 161
6.7.1 Router „verkonfiguriert" oder Passwort vergessen 161
6.7.2 Reparatur „gebrickter" WRTs 163

7 Elektromagnetische Wellen und Antennen 169

7.1 Bel und Dezibel 170
7.2 Das Phänomen der elektromagnetischen Wellen 171
7.3 Das Prinzip von Antennen 173
7.3.1 Polarisation von Antennen 175
7.3.2 Freifelddämpfung und Absorptionsverluste 176

7.4 Antennengewinn 177
7.5 Antennentypen 178
7.5.1 Omni-Antennen 178
7.5.2 Sektorantennen 179
7.5.3 Richtantennen 180
7.6 Wireless Long Shots - Funkstrecken über große Entfernungen . 180
7.6.1 Link Budget - wie weit kann man funken? 180
7.6.2 Signal-Rauschabstand 183
7.6.3 Fresnel-Zone, Multi-Path-Propagation und Phasenrauschen 184
7.6.4 Erdkrümmung 186
7.6.5 Timing-Probleme bei Langstreckenverbindungen 187
7.6.6 Planung von Wireless Long Shots 188
7.7 Sind elektromagnetische Wellen von WiFi-Geräten gefährlich? 188

8 Die Community **191**

Vorwort

Dieses Buch ist, obwohl in manch einsamen Stunden geschrieben, letztlich das Produkt einer weltweiten Community, deren Motto lauten könnte: „Freie Kommunikation für alle!" Dabei ist „frei" im Sinne von Freiheit gemeint. Aktivisten aus mehr als 30 Nationen treffen sich einmal im Jahr zum „Wireless Summit for Free Information Infrastructures"[1] – im vergangenen Jahr in Indien, 2007 in Ghana.

Letztlich habe ich nur aufgeschrieben und dokumentiert, was viele Menschen mit Leidenschaft und Idealismus gemeinsam geschaffen haben. Dazu wurde und wird stetig experimentiert, programmiert, diskutiert und geforscht. Ohne das Internet wäre das Entstandene undenkbar, und es wird Zeit, dass auch die Menschen in weniger entwickelten Regionen dieses kleinen Planeten die Möglichkeit haben, sich durch Wikis, Blogs, Newsserver, Mailinglisten, Chat, IRC, Webserver, VoIP, FTP etc. gegenseitig auszutauschen, zu koordinieren und die Früchte ihrer Arbeit der ganzen Welt zur Verfügung zu stellen. Das Prinzip von Open Source eröffnete den Horizont für Creative Commons und seit einigen Jahren auch für Open Communication mit drahtloser Netzwerktechnologie.

Das größte Potential der globalen Kommunikation via Internet besteht darin, dass sich viele Köpfe zu einem global denkenden Organismus vernetzen können, um Probleme zu lösen und sich weiter zu entwickeln. Heute kommuniziere ich gleichzeitig und praktisch kostenlos mit Menschen aus Nepal, Indien, Bangladesh, den USA, Japan, Lettland – um nur einige zu nennen – und natürlich auch mit den Freifunkern aus Deutschland, der Schweiz und Österreich. Vor einigen Jahren war das noch undenkbar, unbezahlbar und hätte ohnehin eine Ewigkeit gedauert.

Die Diskriminierung durch Armut ist eine der größten Geißeln der Menschheit. Wer sich die gesellschaftliche Teilnahme nicht leisten kann ist von ihr ausgeschlossen. Kommunikation kann nur dann frei sein, wenn sie auch bezahlbar ist oder im besten Fall nichts kostet. Mein Dank geht darum an alle Menschen, die daran arbeiten.

[1] http://wsfii.org

Besonders danken möchte ich ebenfalls dem über alle Maßen tapferen und geduldigen Lektor Ulrich Wolf von Open Source Press, dem meine zahlreichen Versuche, den vorgesehenen Umfang des Buches zu sprengen, und die vielen fehlenden Kommas das eine oder andere graue Haar beschert haben dürften.

Ich möchte außerdem all jenen danken, die unermüdlich an den Routingprotokollen gebastelt haben. Das sind ganz besonders Bruno Randolf, Thomas Lopatic, Axel Neumann und Marek Lindner – und all die anderen Entwickler von OLSR und B.A.T.M.A.N. Vielen Dank auch an Ludger Schmudde für das Webinterface zu B.A.T.M.A.N. und den Mut, immer wieder neue Versionen unserer unausgegorenen Routingsoftware im härtesten Real-Life-Production-Deployment einzusetzen, das ich kenne (auf einem Kirchturm mitten in Berlin-Kreuzberg mit acht Ecken, ebenso vielen Routern und doppelt so vielen Netzwerkinterfaces). Und natürlich an Felix Fietkau, den ich lange wegen der Fehler im Ad-Hoc-Modus des Madwifi-Treibers genervt habe. Vielen Dank auch an die anderen Entwickler von OpenWRT für ihre großartige Arbeit. Dank geht an Sven-Ola Tücke für die Antworten auf viele Fragen und die tolle Firmware und Patches – ohne die Freifunk-Firmware wären wir lange noch nicht da, wo wir heute sind. Vielen Dank an die Teilnehmer, Organisatoren und Sponsoren von WSFII. Dem Chaos Computer Club sei gedankt für die Gelegenheit, Präsentationen über das Thema zu halten, und weil Euer Chaos-Camp der Schauplatz des ersten Meshversuchs mit mehr als zwei Knoten war. Und nicht zuletzt geht ein besonders warmes Dankeschön an Sven Wagner und alle Crewmitglieder der C-Base in Berlin, weil Eure Raumstation ein so angenehmer und kreativer Space ist, in dem sich die Berliner Freifunker jeden Mittwoch zum Wavelöten treffen können. Dem MIT-Roofnet für ETX, den Freifunkern in Nah und Fern, den Community-Netzwerkern überall auf der Welt, dem Verlag Open Source Press und und und...

Eine solche Liste ist entweder endlos oder unvollständig. Seid mir darum nicht böse, wenn Ihr nicht persönlich genannt seid – Ihr und andere wissen, welchen Beitrag Ihr geleistet habt! Vielen Dank für den Tee, das Bier und den ganzen Rest!

Corinna „Elektra" Aichele, im Juli 2007

Kapitel 1

Einleitung

Innerhalb kurzer Zeit nach der Markteinführung der ersten Geräte für Wireless Local Area Networks (WLAN) nach dem IEEE-Standard 802.11 entstanden in verschiedenen Metropolen – zuerst in London, Amsterdam und Berlin – die ersten Bürgernetze auf der Grundlage der neuen Technologie.

Mittlerweile erleben Community-Netzwerke einen globalen Boom. Zwar ist für einen Großteil der Bevölkerung heute ein breitbandiger Internetzugang ebenso selbstverständlich wie Wasser- und Stromanschluss, doch selbst in Industrienationen wie der Bundesrepublik gibt es riesige Lücken in der breitbandigen Internetpenetration, die selbst große Stadtteile in der Hauptstadt Berlin unversorgt lassen.

Schuld an diesem Problem ist u. a. die „Optische Anschlussleitung" (OPAL). Die Glasfaser kam vor allem beim Auf- und Ausbau des Telefonsystems in Ostdeutschland in den 1990er Jahren zum Einsatz und wurde zum Hemmschuh, als die DSL-Technik eingeführt wurde, die auf Kupferleitungen basiert. Für die Betroffenen tut sich also selbst im Land der „Exportweltmeister" der vom ehemaligen UN-Generalsekretär Kofi Annan konstatierte *Di-*

gital Divide, die „Digitale Kluft", auf: Wer keinen Anschluss hat, verliert den Anschluss an die globale Gesellschaft, an Bildung, Kommunikation und Entwicklung. Länder wie Estland wirken dem mit großer Weitsicht entgegen und nehmen eine Vorreiterrolle ein, indem der kostenlose Zugang zum Internet als Grundrecht in der Verfassung verankert ist.

Doch das erklärte Ziel der *Freifunker* – wie sich viele Aktivisten der Community-Netzwerke selbst bezeichnen – ist nicht unbedingt nur das drahtlose Verteilen und Teilen von Internetzugangspunkten. Lokale Inhalte und lokale Dienste sollen an Bedeutung gewinnen.

Bei vielen kommunalen Verwaltungen rennen die freien Netzwerker inzwischen offene Türen ein. Kirchen gestatten den Bürgernetzen Installationen auf Kirchtürmen und lehnen andererseits die Montage von Basisstationen von Mobiltelefonanbietern ab. Mit dazu beigetragen haben Vorbilder wie der Bürgermeister von San Francisco, der in Zusammenarbeit mit Google seine Stadt mit kostenlosem drahtlosen Internetzugang per Mesh-Technologie ausstatten lässt. Eine dazu passende Notiz am Rande ist, dass Google die Firma Meraki Networks aufgekauft hat, die in ihren Wireless Routern das OLSR-Protokoll für Meshnetzwerke einsetzt, welches durch die Berliner Freifunker weiterentwickelt wurde.

Freies Funken und rechtliche Grenzen

Dabei war die Entwicklung von Community-Netzen ein Weg mit Hindernissen. Eine wichtige Voraussetzung war, dass der 2,4-GHz-Frequenzbereich für die lizenzfreie Nutzung durch Privatpersonen zum Aufbau schneller drahtloser Datennetze verfügbar wurde. Bis dahin konnten nur Funkamateure und CB-Funker privat Datenfunkstrecken mit Packet Radio aufbauen. Die Nutzbarkeit von Packet Radio ist jedoch sehr beschränkt. Es ist sehr langsam – vergleichbar mit Modems aus den 1980er Jahren –, und Daten dürfen im Amateurfunk nur zum Zwecke des Hobbys und nur unverschlüsselt übertragen werden.

Die lizenzfreie Datenübertragung mit WLAN war in der BRD zuerst nur innerhalb privater Grundstücke – nicht jedoch über Grundstücksgrenzen hinweg – gestattet. Der Sinn derartiger Beschränkungen ist natürlich zweifelhaft, da elektromagnetische Wellen kaum an Grundstücksgrenzen Halt machen. Die ersten Funkstrecken der deutschen Community-Netzwerker der Initiative Freifunk bewegten sich deshalb in einer rechtlichen Grauzone, um es vorsichtig auszudrücken.

Nach und nach wurden die gesetzlichen Bestimmungen gelockert. In einem nächsten Schritt wurden grundstücksübergreifende Datenverbindungen gestattet, blieben aber meldepflichtig. Dann wurde auch die Meldepflicht aufgehoben. Seit einigen Jahren kann man die Frage, ob ein Community-Netzwerk in Deutschland legal ist, eindeutig bejahen.

Meshrouting vergrößert die Reichweite

Mit verantwortlich für den Boom der Community-Netzwerke ist neben dem drastischen Preisverfall bei WLAN-Geräten das dynamische Meshrouting, das aktiv von der Bürgernetzinitiative Freifunk weiterentwickelt wurde, denn die Reichweite der in Bürgernetzen verwendeten WiFi-Geräte ist an sich gering.

Um über größere Entfernungen miteinander kommunizieren zu können, müssen Daten so lange von Netzwerkknoten zu Netzwerkknoten weitergereicht werden, bis sie am Ziel ankommen. Die Netzwerkteilnehmer müssen dazu ein Maschennetz bilden (engl.: Mesh), bei dem jeder Knoten mindestens einen Nachbarn hat, der für ihn das Weiterreichen von Daten übernehmen kann. Voraussetzung für den Betrieb eines Meshnetzwerks ist Software, die automatisch den günstigsten Weg für das Weiterleiten der Informationen zum Ziel anpasst, wenn neue Netzwerkknoten hinzukommen, sich bewegen oder ausfallen.

Bei einem Community-Netzwerk verläuft der Aufbau üblicherweise anders als in kommerziellen Netzwerken, wo eine Firma in eine zentralisierte Infrastruktur investiert, mit der sich die Endgeräte der Kunden verbinden. In einem Community Netzwerk, das auf Mesh-Technologie setzt, bilden die Geräte der Netzwerkbenutzer selbst einen Teil der Infrastruktur: Das Netz basiert darauf, dass sich die Teilnehmer gegenseitig beim Datentransport helfen. Die Hierarchie Anbieter-Konsument oder Infrastruktur-Endgerät ist in dynamisch „vermaschten" Meshnetzwerken beseitigt.

Aus dem Forschungslabor aufs Hausdach

Die Forschung an der drahtlosen Mesh-Technologie wird seit Beginn der 1970er Jahre im akademischen und militärischen Rahmen betrieben. Funktionierende, frei verfügbare technische Lösungen für Meshnetzwerke gibt es aber erst, seit die Bürgernetzaktivisten die Möglichkeiten der Mesh-Technologie für sich entdeckt haben. Als sie begannen, damit zu experimentieren, erwies sie sich als praktisch kaum entwickelt und in der Praxis unbrauchbar. Doch sie waren von der Idee fasziniert und begannen die Ärmel hochzukrempeln.

Die Community hatte gegenüber der akademischen Forschung den entscheidenden Vorteil, über eine immer weiter wachsende reale Testumgebung zu verfügen, in der man im echten Leben die Ideen und Veränderungen ausprobieren konnte. Die Community hat so das Meshing aus dem Dornröschenschlaf erweckt und gezeigt, dass große dezentrale drahtlose Netzwerke aus vielen einzelnen kleinen Funkzellen möglich sind, die automatisch von intelligenter Software vernetzt werden können.

Eine entscheidende Einschränkung von WLAN-Hardware ist deren kurze Reichweite. Die Sendeleistung ist in Europa auf 0,1 Watt bei 802.11b und 802.11g beschränkt. Die meisten Hersteller geben für ihre SoHO-Geräte eine Reichweite von wenigen hundert Metern im Freien oder weniger als hundert Meter innerhalb von Gebäuden an. Anders als im Amateurfunk, wo es um die direkte Überbrückung möglichst großer Distanzen bei geringer Informationsmenge geht, ist ein Community-Netz auf das Verknüpfen relativ kurzer, breitbandiger Verbindungsstrecken angewiesen. Mit etwas Basiswissen über die Physik der elektromagnetischen Wellen und zusätzlichen Antennen lässt sich die Reichweite jedoch auf mehrere Kilometer steigern, ohne dass die Übertragungsgeschwindigkeit drastisch leidet – auch ohne dabei legale Grenzwerte zu überschreiten.

Fallende Hardwarepreise senken Einstiegshürde

Eine gute Abdeckung mit dem freien Funknetz benötigt eine große Anzahl von Netzwerkknoten, also ein eng verknüpftes „Maschennetz". Durch den Preisverfall bei WiFi-Geräten[1] und den Fortschritt in der drahtlosen Routingtechnologie ist die Zugangsschwelle für Menschen gesunken, die an einem Bürgernetz teilnehmen wollen. Der von Moore beobachtete Effekt, dass sich die Leistungsfähigkeit von Informationstechnologie bei gleichzeitigem Preisverfall beständig steigert, hat erwartungsgemäß auch im Bereich drahtloser Router und Netzwerkkarten keine Ausnahme gemacht.

Im Jahr 2000 kostete ein einfacher Accesspoint von Lucent umgerechnet rund 400 Euro und eine WLAN-Karte für Notebooks vom selben Hersteller 160 Euro. Heute kostet ein WLAN-Router wie der beliebte WRT54GL von Linksys mit eingebautem Ethernetswitch etwa 60 Euro, und preiswerte WLAN-Karten für Notebooks oder Desktop-Rechner sind für weniger als 20 Euro zu haben.

Einige WiFi-Router sind nicht nur günstig, sondern auch sehr flexibel, denn es existieren für diese Geräte aus dem sogenannten SoHO-Bereich (Small or Home Office) hochentwickelte alternative Betriebssysteme auf der Basis von freier Software, die von Open-Source-Entwicklern und Bürgernetzaktivisten entwickelt werden. WiFi-Router dieser Art werden für kleinere Büros entwickelt, oder um das drahtlose Surfen zu Hause auf dem Sofa oder im Garten zu ermöglichen.

Durch freie Software lässt sich das Einsatzspektrum enorm erweitern. Damit werden diese günstigen Geräte zunehmend zur Konkurrenz von Produkten, die für Wireless Internet Service Provider zugeschnitten sind. Der Streckenrekord über eine Distanz von 382 Kilometern wurde mit zwei SoHO-

[1] Die Begriffe WiFi (Wireless Fidelity) und Wireless LAN sind austauschbar, auch wenn WiFi streng genommen nur eine Form eines drahtlosen LANs ist. In diesem Buch werden beide Begriffe synonym verwendet.

Routern für 60 US-Dollar erreicht, die mit der freien Linux-Firmware OpenWRT ausgestattet waren.

Chancen für Entwicklungsländer

In den Entwicklungs- und Schwellenländern verfolgt man die Evolution der Meshtechnologie mit WLAN mit großem Interesse. Die Autorin dieses Buches hat auf der Basis dieser Technologie am Aufbau von Kommunikationsinfrastruktur in Bangladesh und Indien mitgewirkt. In diesen Ländern existieren, abgesehen von GSM, oft keine Kommunikationssysteme, die der Bevölkerung zur Verfügung stehen.

Da in Bangladesh die Bevölkerung überwiegend aus Analphabeten besteht, ist zuerst der Aufbau von Telefonsystemen mit Voice over IP vordringlich. Über eine Funkstrecke mit 6 Megabit Kapazität lassen sich bereits viele Telefongespräche gleichzeitig abwickeln. Antennen, Kabel und Elektronik für diesen Zweck kosten pro Standort nicht mehr als 500 US-Dollar. Unter günstigen Bedingungen können damit leicht 30 km überbrückt werden.

Über dieses Buch

Dieses Buch ist praxisorientiert. Es beschreibt die Hardware und Software, die für die Teilnahme an einem Mesh nötig ist. Softwareseitig liegt der Schwerpunkt auf den Routing-Daemons Olrsd und Batmand sowie der Freifunk-Firmware und OpenWRT, die deren bequeme Installation und Konfiguration in SoHO-Routern ermöglichen.

Um die Software einsetzen zu können, braucht es ein Verständnis von Meshrouting im Allgemeinen und der eingesetzen Routing-Protokolle, die am Anfang des Buches beschrieben werden.

Daneben kommt auch die Hardware nicht zu kurz, denn wenn man seinen Router stabil und mit gutem Datendurchsatz betreiben will, gilt es vieles zu beachten – von der Stromversorgung bis zur Funkreichweite. Das Buch liefert hier zahlreiche Praxistipps und eignet sich deshalb auch als guter Einstieg für alle, die mehr aus ihrer Router-Hardware herausholen wollen.

2 Kapitel

Meshrouting

Wireless LAN nach IEEE 802.11 verfügt über zwei verschiedene Betriebsarten: Den Ad-Hoc-Modus – unter Microsoft Windows als Peer-to-Peer-Modus bezeichnet – und den Infrastrukturmodus. Die meisten Anwender benutzen WLAN im Infrastrukturmodus, wenn sie sich mit einem Accesspoint in einem Hotspot verbinden. Die gesamte WLAN-Kommunikation zwischen den Endgeräten läuft in einem Hotspot über den Accesspoint ab, der üblicherweise als Internetzugangspunkt dient.

Technisch bezeichnet man eine solche sternförmige Netzwerkstruktur als *Point-to-Multipoint-* oder *One-to-Many*-Topologie. Die WLAN-Basisstation verhält sich in einem Hotspot wie der Funkmast eines Mobiltelefonanbieters. Wenn zwei Mobiltelefone miteinander kommunizieren, die sich bei derselben Basisstation eingebucht haben, werden alle zur Abwicklung des Gesprächs notwendigen Daten über den Mobilfunkmast als Relaisstation übertragen.

Befindet man sich in einer abgeschiedenen Gegend ohne funktionierende Infrastruktur, ist das Mobiltelefon nutzlos. Anders als bei einem klassischen

Funkgerät kann man andere Gesprächspartner auch dann nicht anrufen, wenn sich ihre Telefone innerhalb der Funkreichweite befinden.

Abbildung 2.1: WLAN im Infrastrukturmodus

Im Ad-Hoc-Modus hingegen können WLAN-Karten wie Funkgeräte direkt miteinander kommunizieren, ohne auf einen Accesspoint angewiesen zu sein. Jedes Gerät kann gleichzeitig mit mehreren anderen Geräten Verbindungen direkt aufbauen, sofern sie in Funkreichweite sind. Entsprechend wird diese Netzwerktopologie als Multipoint-to-Multipoint oder Many-to-Many bezeichnet.

Abbildung 2.2: WLAN im Ad-Hoc-Modus

Haben alle Teilnehmer in einer Ad-Hoc-Funkzelle untereinander direkten Funkkontakt, wird die Netzwerktopologie als *Full Mesh* (Vollmesh) bezeichnet. Befinden sich zum Beispiel mehrere Notebooks mit WLAN in einem Sitzungssaal, ist die Reichweite der WLAN-Karten üblicherweise ausreichend für ein Vollmesh. Sind die Geräte über einen größeren Bereich verteilt (z. B. Universitätscampus), können Ad-Hoc-Stationen jeweils nur mit den Netzwerkknoten kommunizieren, die sich gerade innerhalb der Funkreichweite befinden.

2.1 Ad-Hoc vs. Infrastruktur

Der Ad-Hoc-Modus hat Vorteile gegenüber dem Infrastrukturmodus - und natürlich auch Nachteile. Ein direkter Vorteil liegt auf der Hand: Da Ad-Hoc-Netze nicht auf eine Basisstation angewiesen sind, fallen beim Betrieb die Kosten für den Accesspoint weg, sofern keine Relaisstation an einem exponierten Standort zur Reichweitenverbesserung erforderlich ist.

Dezentrale Ad-Hoc-Netze sind prinzipiell robuster als zentralisierte Netzwerke im Infrastrukturbetrieb. Fällt die Basisstation in einem Infrastrukturnetz aus, bricht die Kommunikation unter ihren Clients zusammen, in

einem Ad-Hoc-Netz ist der Ausfall eines einzelnen Netzwerkknotens nicht unbedingt störend.

Der direkte Datentransfer zwischen Ad-Hoc-Stationen ist außerdem erheblich schneller als unter Clients im Infrastrukturmodus. Der Transport von Daten zwischen zwei Endgeräten in einem Hotspot setzt mindestens zwei Übertragungsvorgänge voraus (von Gerät A zum Accesspoint, vom Accesspoint zu Gerät B). Zwei Netzwerkknoten (Nodes) können bei der Verwendung von 802.11g im Ad-Hoc-Modus Daten mit einer Geschwindigkeit von maximal 3,2 Megabyte pro Sekunde direkt übertragen. Benutzen beide Geräte den Infrastrukturmodus, reduziert sich die Übertragungsgeschwindigkeit im günstigsten Fall auf 1,6 Megabyte pro Sekunde.

Diese Geschwindigkeit wird im Infrastrukturmodus nur erreicht, wenn beide Notebooks eine hervorragende Übertragungsqualität hin zum Accesspoint haben und keine anderen Geräte diesen gerade benutzen. Verwenden mehrere Endgeräte den Accesspoint gleichzeitig, teilt sich die nutzbare Bandbreite unter allen Teilnehmern auf. Noch drastischer ist der Geschwindigkeitseinbruch, wenn beide Endgeräte zwar nahe zusammen stehen, aber vom Accesspoint entfernt sind, so dass die Übertragung zum Accesspoint ohnehin mit stark reduzierter Datenrate läuft. In diesem Fall schnurrt die nutzbare Bandbreite weiter zusammen.

Natürlich bietet ein fest installierter Accesspoint auch Vorteile. Zweckmäßigerweise wird man einen möglichst hohen und freien Standort wählen und ihn mit einer Antenne ausstatten, welche die Reichweite steigert. Damit vergrößert sich die Reichweite gegenüber Netzwerkknoten, die sich in Bodennähe zwischen Objekten wie Möbeln, Trennwänden und Menschen befinden, welche die Funkübertragung erheblich behindern. Befindet sich Notebook A außerhalb der Übertragungsreichweite von Notebook B, ist im Ad-Hoc-Modus kein Datentransfer zwischen beiden mobilen Netzwerkknotenpunkten möglich, solange kein weiterer Netzwerkknoten als Vermittler zwischen den beiden Geräten agiert. Im Infrastrukturmodus übernimmt der Accesspoint von seinem exponierten Standort aus diese Mittlerrolle und vergrößert die Reichweite und Qualität der Abdeckung.

Die Tatsache, dass ein Accesspoint stets als Funkbrücke arbeitet und deshalb potentiell zur Reichweitenvergrößerung beiträgt, ist aber kein entscheidender Grund gegen den Einsatz des Ad-Hoc-Modus. Auch ein Netzwerkknoten im Ad-Hoc-Modus kann an einem exponierten Standort als Relaisstation betrieben werden. Das Weiterleiten von Daten für andere Netzwerkknoten ist zwar keine Standardfunktion des Ad-Hoc-Modus laut Protokoll IEEE 802.11 a/b/g, diese fehlende Funktion aber kann durch den Einsatz einer Routingsoftware hinzugefügt werden. Dann wird jeder Netzwerkknoten zum Router.

Ein solcher Meshrouter verhält sich dabei weitaus intelligenter als ein herkömmlicher Accesspoint. Dieser wiederholt einfach blind jedes drahtlos

zwischen den Clients übertragene Datenpaket, welches nicht per Kabel ins Internet oder lokale LAN weitergeleitet wird. Mit einem Routingprotokoll ausgestattet, kann jeder einzelne Netzwerkknoten entscheiden, ob das zielgerichtete Weiterreichen von Daten gerade erforderlich ist. Damit steigt die Qualität der Abdeckung, da jedes Gerät im Netzwerk bei Bedarf als Relaisstation arbeitet. Durch den zielgerichteten Transfer verbessert sich die Skalierbarkeit und der mögliche Datendurchsatz der Funkzelle. Außerdem werden drahtlose Datenverbindungen über mehrere Zwischenstationen - sogenannte Multihop-Links - möglich. Durch den Einsatz eines Routingprotokolls wird aus einer simplen Ad-Hoc-Funkzelle ein Meshnetzwerk, das beim Hinzukommen von neuen Netzwerkstationen dynamisch wächst. Jeder neue Teilnehmer verbessert die Funkversorgung und wird dynamisch vom Routingprotokoll in das Netz eingegliedert.

2.1.1 Bandbreitenproblematik bei Multihop-Links mit nur einem Interface

Die meisten Meshknoten in einem Community-Meshnetzwerk arbeiten aus Kostengründen mit nur einer WLAN-Karte. Das reduziert bei einer Route über mehrere Zwischenstationen mit jeweils nur einem drahtlosen Interface die verfügbare Bandbreite bei jedem Weiterreichen der Datenpakete. WLAN-Karten nach 802.11a/b/g arbeiten im sogenannten Halbduplex-Modus. Sie haben einen Empfänger und einen Sender, die auf dem gleichen Kanal arbeiten. WLAN-Karten können also abwechselnd entweder senden oder empfangen - aber nicht beides gleichzeitig. Man kann schließlich nicht zuhören, solange man sich selbst ins Ohr brüllt. Als weiteres Handicap kommt hinzu, dass ein Meshrouter, der gerade Daten an einen anderen überträgt, den Funkkanal für andere Knoten in der unmittelbaren Nähe blockiert.

Überträgt Router A Daten an B, kann C nicht gleichzeitig an D senden, da die Sendung von C den Empfang von B durch Interferenz stören würde.

Reicht B im nächsten Schritt die Daten an C weiter, kann A nicht an B senden, da B gerade selbst sendet. D kann ebenfalls nicht senden, weil er damit den Empfang der Daten von B bei C blockieren würde.

Die Datenrate hat sich bei der Ankunft der Daten bei C bereits halbiert. Bei der anschließenden Übertragung von C an D hört B unfreiwillig mit. A kann

nicht an B senden, da sich im Empfänger von B die Signale von C und A gegenseitig stören würden.

Wenn die Daten schließlich bei D eintreffen, ist die Bandbreite auf dem Pfad A–B–C–D auf rund ein Drittel gesunken.

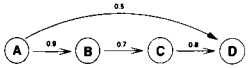
Auf den ersten Hops einer Multihop-Route reduziert sich also die Bandbreite schnell. Sie sinkt jedoch mit weiter ansteigender Sprunganzahl (Hopcount) langsamer und fällt nicht ins Bodenlose (zumindest in der Theorie). Nach fünf Hops stabilisiert sie sich auf niedrigem Niveau.

Diese Aussage trifft jedoch nur unter idealen Bedingungen zu. Die Voraussetzung dafür ist, dass das Mesh sich in einer funktechnisch ruhigen Umgebung befindet und die Übertragungen nicht von anderer Seite gestört werden. In dicht bevölkerten Gebieten ist das im 2,4-GHz-Bereich mit seinen zahlreichen konkurrierenden Anwendungen jedoch kaum der Fall.

Es ist eine typische Eigenschaft der „Luftschnittstelle", dass sich die Bedingungen sprunghaft ändern können. In einer Großstadt führt das dazu, dass auf einer Multihop-Route mit vielen Hops häufig mindestens eine Übertragungsstrecke gerade stark gestört ist und praktisch kaum Bandbreite zur Verfügung stellt. Eine Kette ist nur so stark wie ihr schwächstes Glied – da jede Funkstrecke individuell durch Interferenzen gestört werden kann, steigt mit dem Hopcount die Wahrscheinlichkeit, dass eine Route augenblicklich nicht funktioniert.

Mit steigendem Hopcount sinkt daher nicht nur die Bandbreite – gleichzeitig steigen auch die Paketverluste und die Wahrscheinlichkeit, dass eine Route gerade nicht passierbar ist, bevor das Routingprotokoll eine Alternativroute zur Verfügung stellen kann (sofern sich eine Alternativroute aufbauen lässt).

Paketverluste haben die unangenehme Eigenschaft sich zu multiplizieren. Die Erfolgsrate bei der Übertragung ist in der Grafik als Faktor angegeben. Ein Faktor von 1 entspricht einer Erfolgsrate von 100 Prozent. Durch die Multiplikation der einzelnen Erfolgsraten ergibt sich der Gesamtwirkungsgrad der Multihop-Route.



Multipliziert man den Gesamtwirkungsgrad der Route mit der theoretischen Bandbreite, kommt man zu einer einigermaßen realistischen Einschätzung des tatsächlichen Durchsatzes (Throughput). Eine Route A–B–C–D mit einer theoretischen Bandbreite von 3,66 Megabit und einem Gesamtpaketverlust von 50 Prozent wird einen Durchsatz von etwa 1,8 Megabit bei der Verwendung des verbindungslosen UDP-Protokolls erzielen.

Diese Betrachtung ist allerdings stark vereinfacht. In diesem Beispiel wurde nur der Weg in eine Richtung berücksichtigt und bei allen Verbindungsstrecken die gleiche Bruttodatenrate angenommen. Außerdem hat das jeweils verwendete Protokoll (TCP, UDP) einen nicht unerheblichen Einfluss auf die erzielbare Bandbreite.

2.1.2 Problemfall TCP

Im Gegensatz zum verbindungslosen Protokoll UDP ist TCP (Transfer Control Protocol) ein verbindungsorientiertes Protokoll, welches überprüft, ob die Datenübertragung zwischen zwei Netzwerkknoten erfolgreich stattgefunden hat. Der Sender wartet nach dem Verschicken eines Datensegments auf eine Empfangsbestätigung (Acknowledgement) des Empfängers, in der dieser mitteilt, ob er die Daten fehlerfrei empfangen hat. Trifft innerhalb der Wartezeit (Timeout) das Acknowledgement für das übertragene Datensegment nicht ein, wird das Datenpaket als verloren betrachtet und erneut übertragen. Die Performance von TCP ist in einem Mesh sehr schlecht, da das Protokoll ausschließlich für die Anforderungen kabelgebundener Netzwerke entwickelt wurde, bei denen Paketverluste nur selten vorkommen.

In Kabelnetzen tritt Paketverlust nicht wie beim Funk im Übertragungsmedium auf, sondern durch Datenstaus am Übergang zwischen unterschiedlichen Netzwerksegmenten. Eine primäre Zielsetzung von TCP ist es, die Steuerung der Datenrate in einem Netzwerk so zu koordinieren, dass es nicht an Engstellen zu Datenstaus kommt. Würde TCP trotz Datenstau die Übertragung vermeintlich verlorengegangener Pakete in schneller Folge wiederholen, wäre das verheerend – wie ein Stau auf der Autobahn, in den immer mehr Autos hineinfahren.

Wird TCP in einem Funknetzwerk eingesetzt, interpretiert es die Paketverluste durch Übertragungsfehler im Funkmedium fälschlicherweise als Datenstaus und reagiert darauf durch eine Reduzierung der Datenrate. Paketverlusten begegnet es zwar mit erneuten Übertragungsversuchen, es verdoppelt aber bei jeder Retransmission die Wartezeit vor dem erneuten Senden (Exponential Backoff) bis zu 64 Sekunden.

Nach 13 gescheiterten Versuchen oder 9 Minuten gibt TCP auf und erzeugt eine Fehlermeldung. Durch den Exponential Backoff wird die Datenübertragung über TCP auf schlechten Funkstrecken quälend langsam.

Retransmissions von TCP sollten so gut wie es geht vermieden werden. Eine naheliegende Möglichkeit, die Probleme mit TCP abzuschwächen, ist es, die Übertragung unterhalb der TCP/IP-Protokollschichten auf dem Transportlayer der WLAN-Karten so robust wie möglich zu machen.

Wireless LAN nach 802.11 beherrscht die Option *Fragmentation*, die größere Datenpakete bei der Übertragung zwischen zwei WLAN-Karten in kleinere Fragmente unterteilt. Ein großes Datenpaket benötigt in der „Luftschnittstelle" eine länger dauernde, ungestörte Übertragungszeit als ein kleineres Paket und hat damit eine geringere Wahrscheinlichkeit, erfolgreich übertragen zu werden.

Wie bei TCP erwartet die sendende WLAN-Karte nach jedem Datenfragment ein Acknowledgement des Empfängers. Bleibt dieses aus, wird die Übertragung jedes einzelnen Datenfragments wiederholt - üblicherweise bis zu sieben Mal. Diese Funktion sollte in einem Mesh immer verwendet werden. Auf diese Weise lässt sich eine gewisse Übertragungssicherheit erreichen, um zu vermeiden, dass TCP Maßnahmen gegen vermeintliche Datenstaus ergreift. Die beste Erfahrung hat die Autorin mit sehr kleinen Werten für die Fragmentierung gemacht. Die Mindestgröße ist 256 Byte.

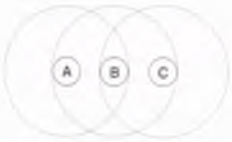
Maximale IP-Paketgröße - Maximum Transfer Unit

In drahtgebundenen Netzwerken wird bei der Konfiguration der Netzwerkschnittstelle der größtmögliche Wert für die Größe der IP-Pakete eingestellt, um einen möglichst großen effektiven Datendurchsatz zu erzielen. Bei jedem IP-Paket müssen zusätzliche Informationen für die Transportschicht als „Verpackung" (Overhead) zu dem zu transportierenden Inhalt (der Payload) hinzugefügt werden. Um das Verhältnis von IP-Overhead und Payload günstiger zu gestalten, wird die Paketgröße daher maximiert. Sowohl bei Ethernet als auch bei WiFi beträgt die Maximum Transfer Unit (MTU) deshalb normalerweise 1500 Byte.

Bei Routen mit vielen Sprüngen (Hops) sollte die oben beschriebene Fragmentierung auf der MAC-Ebene bei allen Karten eingeschaltet sein. Haben die Betreiber von Meshknoten diese Einstellung nicht vorgenommen, ist das Verkleinern von IP-Paketen, also eine kleinere MTU, eine Notlösung, um die negativen Effekte von TCP abzuschwächen. Der positive Effekt ist hier jedoch bei weitem geringer als bei Verwendung der Fragmentierung auf MAC-Ebene. Das Verkleinern der MTU kostet verglichen mit der Fragmentierung einen größeren Teil der erzielbaren Datengeschwindigkeit. Ist aber eine Route sehr wenig belastbar, ist es sinnvoll, zusätzlich zur Fragmentierung auch die MTU zu verkleinern.

2.1.3 Exposed Nodes - Hidden Nodes

Es wurde in den Grafiken zu den Multihop-Transfers stillschweigend davon ausgegangen, dass sich die Knoten A–B–C–D einander nicht dauernd chaotisch dazwischenfunken. Das ist jedoch nicht selbstverständlich.



Hier kann A nur B sehen und C nur B. Aus der Sicht von A ist C ein versteckter Knoten (*Hidden Node*) und ebenso ist A für C unsichtbar. B ist der exponierte Knoten (*Exposed Node*) dazwischen, der alle beide Knoten sieht. Daraus ergibt sich das Problem, dass B Schwierigkeiten hat, Pakete zu empfangen, wenn A und C gleichzeitig senden. B muss koordinieren, welcher von den beiden versteckten Knoten gerade senden darf, so dass sich die Übertragungen von A und C nicht gegenseitig blockieren. Da Meshnetzwerke auf dem Weiterreichen von Daten über Zwischenstationen basieren, ist das Problem mit den versteckten Knoten der Normalfall.

Ganz lassen sich Kollisionen in dieser Situation nicht vermeiden, sie lassen sich jedoch prinzipiell stark vermindern. Bevor Knoten A anfängt, große Datenmengen zu senden, fragt er mit einer kurzen Übertragung per Broadcast, ob er senden darf, dem *Request-to-Send* (RTS). In dem RTS-Datenpaket ist angegeben, wie lange Knoten A das Übertragungsmedium für sich reservieren möchte. Wenn B der Meinung ist, dass A senden darf, funkt er *Clear-to-Send* (CTS). C hört das RTS von A nicht, wohl aber, dass B einer versteckten Station das OK zum Senden gegeben hat, und hält sich für die angeforderte Zeitspanne zurück. Möchte C senden, fragt er ebenfalls vorher per RTS-Broadcast.

Auch der RTS/CTS-Mechanismus ist eine Funktion von WLAN-Karten, die in einem Mesh eigentlich verwendet werden sollte. Die Implementierung von RTS/CTS in IEEE 802.11abg wurde aber nicht für den Einsatz in Meshnetzwerken entwickelt und ist dafür ungeeignet. Sie schadet viel mehr, als sie nützt. In einem Mesh blockieren sich die Knoten mit den RTS/CTS-Signalen häufig irrtümlicherweise gegenseitig. Es kommt zu Situationen, in denen gar kein Knoten sendet, weil alle Knoten fälschlicherweise annehmen, dass das Medium für einen nicht existierenden Knoten reserviert wurde. Es existiert ein Lösungsvorschlag[1] für dieses Problem, der aber in

[1] Saikat Ray, Jeffrey B. Carruthers and David Starobinski: „RTS/CTS-Induced Congestion in Ad Hoc Wireless LANs", `http://ieeexplore.ieee.org/iel5/8546/27030/01200611.pdf`

802.11abg nicht zur Verfügung steht. Deshalb sollte bis auf weiteres auf RTS/CTS im Mesh verzichtet werden.

Empfängt Knoten E ein RTS-Signal, welches nicht an ihn gerichtet ist, stellt er für die im RTS-Paket angegebene Zeitdauer jegliche Aussendungen ein, da für ihn das Medium mutmaßlich blockiert ist. Er tut dies auch dann, wenn er in der aktuellen Situation kein Hidden Node ist und die Übertragung überhaupt nicht stören würde. Wird E in dem Zeitraum, in dem für ihn das Senden blockiert ist, von einem weiteren Knoten F mit einem RTS-Signal nach Sendezeit gefragt, darf er nicht antworten. Da E nicht antwortet, vermutet F eine Kollision und stellt seinerseits für eine Zeit (Backoff-Time) den Sendebetrieb ein. Das RTS-Paket von F veranlasst aber weitere Knoten, die das RTS-Signal empfangen, ebenfalls dazu, für die im RTS-Paket angegebene Zeitdauer ihre Aussendungen einzustellen. Diese Antworten dann ihrerseits auf die RTS-Anfragen anderer Netzwerkkonten nicht. Dieser Irrtum kann sich über das gesamte Mesh ausbreiten, so dass überhaupt kein Knoten mehr sendet!

Im Extremfall tritt das Phänomen unter Meshknoten auf, die einen Kreis bilden. Die RTS/CTS-Blockade pflanzt sich ringförmig fort und kann zu einer Blockade (Deadlock) führen, die einige Zeit anhalten kann.

2.2 Routingprotokolle für drahtlose Netzwerke

Die Entscheidung, auf welchem Weg Daten von jedem einzelnen Netzwerkknotenpunkt zu jedem anderen Node im Netz geroutet werden, kann bei kleinen statischen Netzwerken noch von Hand in die IP-Routingtabellen jedes Netzknoten eingegeben werden. Doch selbst in kleinen drahtgebundenen Netzwerken gerät das schnell zu einem mühsamen und komplizierten Unterfangen. Kommt ein neuer Router hinzu oder fällt einer aus, müssen unter Umständen sämtliche Routingtabellen in allen Routern von Hand bearbeitet werden. Ein derartiges Verfahren wäre viel zu aufwendig und fehlerträchtig. Deswegen arbeitet man in professionellen Netzwerken schon lange mit Software, die Routingtabellen automatisch erzeugt und anpasst.

Für drahtgebundene Netze entwickelte Routingprotokolle eignen sich jedoch nicht für drahtlose Netzwerke. Diese sind permanenten Veränderungen unterworfen, auch wenn sie nur aus statischen Netzwerkknoten auf Hausdächern bestehen, zwischen denen keine die Funkübertragung störenden Objekte auftauchen und wieder verschwinden. Die verfügbare Bandbreite zur Datenübertragung ist prinzipiell eher gering. Ein Routingprotokoll, welches selbst große Datenmengen erzeugt, ist auf schnellen Datenleitungen akzeptabel, solange es zuverlässig funktioniert. In drahtgebundenen Netzen lässt sich die Bandbreite prinzipiell durch mehr Leitungen erweitern, in einem drahtlosen Netzwerk hingegen teilen sich alle Netzwerkknoten mit dem Funkspektrum ein gemeinsames Übertragungsmedi-

um, die Bandbreite ist begrenzt; Routingprobleme lassen sich nicht durch mehr Bandbreite kompensieren.

Von einer kabelgebundenen Datenverbindung kann man weiterhin erwarten, dass das, was man am einen Ende hineingibt, auch auf der anderen Seite herauskommt, solange die Hardware nicht beschädigt ist. Ist eine kabelgebundene Übertragungsstrecke defekt, fällt sie in der Regel komplett aus. Eine Funkstrecke kann zwischen fehlerloser Übertragungsqualität bei hoher Geschwindigkeit oder geringstem Datendurchsatz mit unregelmäßigen Aussetzern schwanken. Derartige Probleme treten in der Praxis in einer Umgebung mit konkurrierenden Funkanwendungen häufig auf.

WiFi arbeitet in Frequenzbereichen, die auch von vielen anderen Funkanwendungen genutzt werden. Störungen durch Interferenzen sind daher eher die Regel als die Ausnahme. Nicht zuletzt erzeugt auch jede Datenübertragung in einem Meshnetzwerk selbst ein Störsignal bei Netzwerkknoten in der Umgebung und schränkt damit die Qualität der anderen Funkverbindungen ein. Die Störreichweite einer Funkübertragung ist mindestens dreimal so hoch wie die Nutzreichweite.

Ein großes, drahtloses Meshnetzwerk ist – vor allem in einer Großstadt – hochgradig dynamischen Veränderungen unterworfen und chaotisch. Wenn ein Knoten ausfällt, kann sich das Protokoll nicht darauf verlassen, dass diese Veränderung zeitnah oder zuverlässig mitgeteilt wird. Ein Routingprotokoll für Meshnetzwerke muss daher robust gegenüber fehlenden oder unzuverlässigen Informationen sein, schnell auf Veränderungen reagieren und mit einem Minimum an selbsterzeugtem Datenverkehr auskommen.

Die Forschungen an Routingprotokollen für Meshnetzwerke laufen seit dem Beginn der 1970er Jahre. Die englischsprachige Wikipedia listet über 70 verschiedene Protokollentwürfe auf[2]. Viele davon existieren nur auf dem Papier oder werden bislang nur in Simulationen eingesetzt. Routingprotokolle für Meshnetzwerke werden in folgende Gruppen eingeteilt:

- proaktive Protokolle
- reaktive Protokolle
- hybride Protokolle (proaktiv/reaktiv)
- hierarchische Protokolle
- geografische Protokolle
- energieorientierte Protokolle
- Multicast-Protokolle
- geografische Multicast-Protokolle

[2] `http://en.wikipedia.org/List_of_ad-hoc_routing_protocols`

2.2.1 Protokolltypen

Proaktive Protokolle ermitteln permanent alle Routen zwischen allen Nodes in einem Funknetz mit einem mathematischen Algorithmus, der den kürzesten (oder günstigsten) Pfad zwischen einem Start- und einem Zielpunkt berechnet. Jeder Knoten berechnet Routen zu allen anderen Knoten im Netzwerk, unabhängig davon, ob diese gerade benötigt werden oder nicht. Sie erzeugen deshalb prinzipbedingt schon in relativ kleinen Netzwerken viel Rechenlast und protokollspezifischen Datenverkehr. Die meisten proaktive Routingprotokolle verwenden den *Shortest-Path-Algorithmus*, der nach seinem Erfinder Edsger W. Dijkstra *Dijkstra-Algorithmus* oder auch *Link-State-Algorithmus* genannt wird. Dazu flutet jeder einzelne Router regelmäßig das gesamte Netzwerk mit Nachrichten, in der er seine lokalen, direkt erreichbaren Nachbarknoten auflistet. Diese Informationen über die lokale Topologie jedes einzelnen Node wird von jedem anderen Teilnehmer in einer eigenen Datenbank gesammelt und zu einem großen Graphen verarbeitet, der das ganze Netz abbildet. Link-State-Verfahren sind in der Lage, bei ihren Routingentscheidungen auch Faktoren wie Bandbreite, Zuverlässigkeit von Verbindungsstrecken, Delay und Auslastung der Verbindungsstrecken zu berücksichtigen.

Selten wird bei proaktiven Protokollen zur Routenberechnung das Distanz-Vektor-Verfahren angewendet. Dazu müssen die einzelnen Knoten ihre Routingtabellen (komplett oder partiell) über das Netzwerk kommunizieren. Dieses Verfahren erzeugt ein noch größeres protokollspezifisches Datenaufkommen als das Link-State-Verfahren und kann sich nur langsam an Veränderungen im Netzwerk anpassen (konvergieren).

Zu den proaktiven Protokollen gehören DSDV (Destination-Sequenced Distance-Vector)[3], B.A.T.M.A.N. (Better Approach to Mobile Ad-Hoc Networking)[4] und OLSR (Optimized Link State Routing)[5].

Reaktive Protokolle ermitteln Routen nur bei Bedarf. Möchte ein Knoten mit einem anderen Netzteilnehmer kommunizieren, flutet er das Netz mit einer Anfrage nach einer Route zu seinem Ziel. Reaktive Verfahren erzeugen nur wenig Protokolldatenverkehr im Netz und verursachen wenig Rechenlast, wenn nur selten Routen gesucht werden und wenn die gefundenen Routingpfade stabil sind. Solange jedoch nicht nach Routen im Mesh gesucht wird, weiß ein Knoten nicht, ob überhaupt andere Knoten im Netzwerk vorhanden oder erreichbar sind.

Da Routen erst bei Bedarf gesucht werden, stehen Netzwerkverbindungen erst nach einer Verzögerung zur Verfügung. Bekannte reaktive Protokolle

[3] `http://citeseer.ist.psu.edu/perkins94highly.html`
[4] `http://open-mesh.net/batman`
[5] `http://hipercom.inria.fr/olsr/`

sind DSR (Dynamic Source Routing)[6], AODV (Adhoc On-Demand Distance Vector)[7], DYMO (Dynamic MANET On-demand)[8] und SrcRR[9].

Hybride Protokolle verwenden bis zu einem bestimmten Horizont ein proaktives und für weiter entfernte Knoten ein reaktives Verfahren. Dadurch sollen die Vorteile von proaktiven Verfahren (Routen sind proaktiv in der Routingtabelle vorhanden) und reaktiven Verfahren (weniger Protokolldatenaufkommen und Rechenaufwand) kombiniert werden, ohne die Nachteile zu übernehmen. Hybride Protokolle sollen für sehr große Meshnetzwerke geeignet sein. Vertreter dieser Gattung sind HSLS (Hazy Sighted Link State Routing)[10] und ZRP (Zone Routing Protocol)[11].

Hierarchische Protokolle versuchen, Netzwerkknoten zu hierarchisch gegliederten Gruppen zusammenzufassen um das Aufkommen von Protokolldatenverkehr zu reduzieren und dadurch die Skalierbarkeit zu erhöhen. Ein Protokoll dieser Gattung ist DART (Dynamic Address Routing)[12].

Geografische Protokolle treffen ihre Routingentscheidungen nach der geografischen Position der Netzwerkknoten. Dieses Verfahren erfordert zusätzliche Hardware zur Positionsbestimmung oder die manuelle Eingabe des Standorts. Nicht berücksichtigt wird dabei, dass sich zwei Knotenpunkte zwar geografisch nahe sein können, die Funkverbindung aber möglicherweise durch Hindernisse völlig getrennt ist.

Energieorientierte Protokolle spielen vor allem für Sensornetzwerke eine Rolle, wo die Einsparung von Sendeenergie optimiert werden soll. Datenpakete sollen nicht mit großer Sendeenergie über längere Strecken übertragen werden, sondern über viele Zwischenstationen, um Sendeenergie zu sparen oder Sensoren schwerer auffindbar zu machen.

Multicast-Protokolle dienen der Übertragung von Daten über verbindungslose Protokolle. Interessant sind sie für Multimedia-Inhalte, etwa Audio- und Videostreams.

Geografische Multicast-Protokolle sollen multimediale Inhalte per Multicast über ein Mesh an Geräte verbreiten, die sich in einem bestimmten geografischen Gebiet befinden. Sie können unter anderem für sogenannte Location Based Services (Standortbezogene Dienste) eingesetzt werden.

[6] David B. Johnson, „Routing in Ad Hoc Networks of Mobile Hosts", Proceedings of the Workshop on Mobile Computing Systems and Applications, pp. 158-163, IEEE Computer Society, Santa Cruz, CA, December 1994.

[7] `http://moment.cs.ucsb.edu/AODV/aodv.html`

[8] `http://moment.cs.ucsb.edu/dymo`

[9] `http://pdos.csail.mit.edu/roofnet/doku.php?id=design&s=srcr#routing_protocol`

[10] `http://www.ir.bbn.com/documents/techmemos/TM1301.pdf`

[11] `http://tools.ietf.org/id/draft-ietf-manet-zone-zrp-04.txt`

[12] `http://dart.cs.ucr.edu`

2.2.2 Praktische Relevanz der Routingprotokolle

Auf dem Gebiet der Routingprotokolle für drahtlose Netzwerke wird international viel geforscht. Es existieren viele theoretische Protokollentwürfe. Doch die meisten Arbeiten dienen in der akademischen Forschung lediglich als Diskussionsgrundlage und werden nie in der Praxis eingesetzt oder getestet, da nur wenige Forschungseinrichtungen die Möglichkeiten für umfangreiche praktische Erprobung haben. Wenn Programmcode für eines der Protokolle existiert, handelt es sich oft nur um Implementierungen für Netzwerksimulatoren, was häufig kritisiert wird.

Es ist nämlich sehr schwer, die komplexe Situation eines realen drahtlosen Netzwerks durch Simulation effektiv nachzubilden. Jedes von einem Router ausgesandte Datenpaket hat Auswirkungen auf andere Datenübertragungen, die gerade im Mesh stattfinden, da die Störreichweite eines ausgesandten Signals schwer zu berechnen ist. Um ein Mesh mit 1000 Meshroutern zu simulieren, müsste das Programm für viele aufeinander folgende Zeiteinheiten von sehr kurzer Dauer jeweils berechnen, was gerade im gesamten Netz passiert, und dabei alle signifikanten Wechselwirkungen berücksichtigen. Dafür ist komplexe Software und eine leistungsfähigen Großrechenanlage nötig.

Eine brauchbare Simulationssoftware ist möglicherweise noch bedeutend schwieriger zu entwickeln als ein wirklich gutes Routingprotokoll. Es liegt daher nahe, die Ideen in einem tatsächlich exisitierenden und genutzten Netzwerk einem Praxistest unter realen Bedingungen zu unterziehen. So können stichhaltige Ergebnisse erzielt werden.

Das MIT (Massachusetts Institute of Technology) gehört mit dem Projekt Roofnet zu den Ersten, die dem Problem durch praktisches Experimentieren in einem realen, von Anwendern genutzten Meshnetzwerk nachgegangen sind. Aus diesem Projekt sind das reaktive Routingprotokoll SrcRR (eine DSR-Variante mit einer Metrik zur Linkbeurteilung, die den Link-State-Algorithmus von Edsger W. Dijkstra verwendet) und die Firma Meraki hervorgegangen[13]. Meraki vertreibt preiswerte Wireless Router auf der Basis von Open-WRT mit installiertem OLSR und SrcRR. Bislang hat die Lösung von Meraki in Europa kaum Verbreitung gefunden.

Die Entwicklung von Protokollen für Meshnetzwerke ist nicht einmal ansatzweise abgeschlossen und bietet ein weites Feld für Forschung und Entwicklung. Dabei werden unterschiedlichste Ansätze verfolgt. In der akademischen Diskussion spielt vor allem die Skalierbarkeit eine Rolle – wie viele Netzwerkknoten sind möglich, bevor die vom Routingprotokoll erzeugte Rechen- und Protokolldatenlast die Netznutzung zu sehr einschränkt.

[13] John Bicket et al., „Architecture and Evaluation of an Unplanned 802.11b Mesh Network", `http://pdos.csail.mit.edu/papers/roofnet:mobicom05/roofnet-mobicom05.pdf`

Aus der – rein subjektiven – Sicht der Autorin ist diese Frage aber von untergeordneter Rolle. Denn der Datenverkehr über eine drahtlose Multihop-Route ist ohnehin nur bis zu einer bestimmten Anzahl von Sprüngen sinnvoll und möglich. Deshalb sind Überlegungen, in einem Mesh zigtausende Router miteinander kommunizieren zu lassen, wenig praxisrelevant. Entscheidend ist viel eher, ob ein Routingprotokoll die günstigsten Routen findet sowie schnell und richtig auf Veränderungen im Netzwerk reagiert.

Das ideale Protokoll sollte immer die bestmögliche Route zur Verfügung stellen und bei einer Störung sofort reagieren. Trotz aller berechtigter Kritik am größeren Rechenleistungs- und Bandbreitenverbrauch sind proaktive Protokolle in der Praxis von größerer Relevanz als reaktive Protokolle, bei deren Entwicklung die Skalierbarkeit im Vordergrund steht. Erstere haben den Vorteil zu funktionieren und sinnvolle Routen zu finden – während letztere zwar theoretisch skalieren, aber in der Realität kaum etwas Sinnvolles zu Stande bringen.

Die Resultate von reaktiven Routingprotokollen sind in der Praxis entweder bedeutend schlechter oder sie enthalten wie das vom Roofnet-Projekt verwendete DSR/SrcRR proaktive Elemente, die ihre Skalierbarkeit zwar reduzieren, sie aber gleichzeitig praxistauglich machen. Rein reaktive Protokolle finden bestenfalls irgendeine Route, welche die kürzeste Verbindung zwischen zwei Knoten darstellt. Soll dagegen die Qualität der Verbindungen bei der Routenfindung eine Rolle spielen, kommen die Protokollentwickler um ein proaktives Messen der Verbindungsqualität der einzelnen Linkstrecken nicht herum.

Alle heute in der Praxis relevanten Routingprotokolle für Meshnetze verwenden eine Metrik, die die Qualität der Funkstrecken berücksichtigt. Entweder ist sie im Algorithmus inhärent wie bei B.A.T.M.A.N. oder sie wird explizit gemessen und permanent überwacht wie bei der OLSR-Implementierung von Freifunk, bei HSLS und SrcRR, der DSR-Implementierung des MIT. Protokolle, die einfach die kürzeste Verbindung zwischen zwei Punkten suchen, sind in der Praxis irrelevant.

Von praktischer Bedeutung für freie Funknetze sind in Europa bislang OLSR (Optimized Link State Routing) und B.A.T.M.A.N. (Better Approach to Mobile Ad-Hoc Networking) aus der Gruppe der proaktiven Protokolle. HSLS (Hazy Sighted Link State Routing) aus der Gruppe der hybriden Protokolle und das reaktive Srcr spielen vor allem in den USA eine Rolle. DSR, AODV, DART, DYMO und ZRP sind dagegen (bislang) irrelevant.

HSLS ist theoretisch wegen seines hybriden Protokollaufbaus für den Einsatz in extrem großen Meshnetzwerken geeignet. Dieses Protokoll wird von CuWIN (Champaign-Urbana Community Wireless Network), einer US-amerikanischen Community-Netzwerk-Initiative implementiert und weiterentwickelt, in Europa ist HSLS nicht verbreitet. Die größten HSLS-Meshes sind mit 25 Knoten (Stand Oktober 2006 laut Chefentwickler David Young) ver-

gleichsweise klein. Bislang gibt es für HSLS nur die Softwareimplementierung von CuWIN, die ausschließlich unter NetBSD arbeitet. CuWIN setzt bei der Hardware auf PCs, die von einer Live-CD mit dem NetBSD-basierten CuWIN-System booten. Dies dürfte eine Ursache für die geringe Verbreitung sein, da die in Community-Netzwerken verwendeten SoHO-Accesspoints üblicherweise Linux verwenden.

Für das SrcRR-Protokoll gibt es bisher keine Implementierung in einer performanten Programmiersprache wie C oder C++. Das MIT Roofnet verwendet die Software *Click Modular Router*, die einen Softwareinterpreter für Routinganweisungen darstellt. Routingabläufe werden in der Hochsprache des Click Modular Router als Programmmodule formuliert und diesem beim Start zur Ausführung übergeben. Click ist eine hervorragende Umgebung für die Entwicklung von Netzwerkdesigns, da der Click Modular Router sowohl auf Computern als auch im Netzwerksimulator NS2 lauffähig ist. Der für Click geschriebene Protokollcode kann daher ohne Veränderungen im Simulator oder in einem realen Netzwerk getestet werden. Diese Vorgehensweise ist für die Entwickung von Protokollen sehr elegant, bedingt aber auch einen erhöhten Rechenaufwand beim Einsatz eines Routingprotokolls, der Probleme mit der Skalierbarkeit in einem Mesh aufwirft, wenn Consumer-Accesspoints wie der Linksys WRT54G beziehungsweise WRT54GL eingesetzt werden. Dessen ungeachtet verkauft die Firma Meraki kleine Meshrouter mit der Leistungsfähigkeit von SoHO-Geräten, in denen sowohl Click als auch OLSR installiert ist.

OLSR und B.A.T.M.A.N. werden in den folgenden Kapiteln ausführlich vorgestellt.

Kapitel 3

Optimized Link State Routing

Optimized Link State Routing (OLSR) gehört zu den proaktiven Routingprotokollen, die mit dem Ziel entwickelt wurden, den Zustand des gesamten Netzwerks kontinuierlich zu erfassen. In jedem Meshknoten werden ständig Routen zu allen anderen erreichbaren Netzwerkknoten berechnet und bereitgehalten.

Um das zu erreichen, prüft jeder Knoten in regelmäßigen Intervallen durch das Versenden und Empfangen von Hello-Nachrichten, ob sich Nachbarn in direkter Funkreichweite befinden. Diese Informationen über die lokalen Nachbarschaftsverhältnisse – also welcher Knoten welche Nachbarn hat – werden immer wieder über das ganze Netzwerk verteilt. Jeder Knoten sammelt sie in seiner eigenen Datenbank, die das gesamte Netzwerk abbildet. Auf diese Datenbank wird beim Eintreffen jeder neuen Nachricht über den Zustand des Netzwerks der *Shortest Path Algorithm* angewendet, um die kürzeste Verbindung zwischen allen Netzwerkknoten zu berechnen.

Das Vorgehen entspricht einem klassischen Link-State-Algorithmus. Diese erzeugen jedoch sehr hohe Rechenlast und viele Protokolldaten, die die

Nutzbandbreite des Netzwerks reduzieren. Ziel von OLSR ist es, das klassische Link-State-Routing im Hinblick auf Skalierbarkeit und Zuverlässigkeit zu optimieren. Zu diesem Zweck setzen die OLSR-Entwickler auf die Erweiterung des Link-State-Routings um sogenannte Multipoint-Relais, die das Fluten des Netzwerks mit Topologieinformationen effektiver gestalten sollen. Diese Technik wird im Abschnitt 3.2.2 näher erörtert.

3.1 Entwicklungsgeschichte

OLSR wurde als Internet Draft RFC 3626 im Jahr 2003 von Mitarbeitern der staatlichen französischen Forschungseinrichtung INRIA bei der IETF eingereicht. Es wird unter anderem von der US-Navy zur Kommunikation zwischen Kriegsschiffen eingesetzt. Für OLSR nach RFC 3626 gibt es zahlreiche unterschiedliche Implementierungen – unter anderem von der INRIA, der US-Navy und Olsr.org.

Die OLSR-Implementierung von Olsr.org wurde von Andreas Toennesen im Rahmen seiner Diplomarbeit an der Universität Oslo erstellt. Sie ist Open Source und steht unter der sehr liberalen BSD-Lizenz. Das OLSR-Projekt von Andreas Toennesen hat sich in Zusammenarbeit mit den Freifunkern im Laufe der Jahre weiterentwickelt. Die derzeitige OLSR-Variante hat zwar ihre Wurzeln in RFC 3626, sie ist aber nicht mehr interoperabel zum Entwurf der INRIA. OLSR-Knoten, die nach RFC 3626 oder Freifunk-OLSR arbeiten, ignorieren sich gegenseitig. Die von Olsr.org erhältliche Routingsoftware lässt sich aber so konfigurieren, dass sowohl ein RFC-konformes Verhalten als auch der Betrieb in einem Freifunk-Mesh möglich ist.

Die von den Freifunkern modifizierte Variante von OLSR ist in freien Funknetzen mit bis zu 600 Knotenpunkten weltweit verbreitet. Genaue Studien über die Verbreitung gibt es nicht, es dürfte jedoch die mit Abstand am weitesten verbreitete Variante des OLSR-Routingprotokolls sein.

Im Jahr 2004 wurde die OLSR-Implementierung von Andreas Toennesen von den Berliner Freifunkern auf der Konferenz „Wizards of OS 3" in einem Feldversuch mit Konferenzteilnehmern getestet und danach im Berliner Freifunknetz eingeführt. Die Performance war jedoch bei den ersten praktischen Tests keineswegs zufriedenstellend und nicht besser als die des zuvor eingesetzten proaktiven Protokolls Mobilemesh. Die Routingtabelle in den OLSR-Knoten baute sich nur sehr langsam auf und brach ständig ganz oder teilweise zusammen. Die vom Protokoll gewählten Routen waren äußerst instabil und hatten nur eine minimale nutzbare Bandbreite.

Es stellte sich heraus, dass die vorgeschlagenen Optimierungen bis zu diesem Zeitpunkt nur theoretisch oder in Simultationsverfahren auf ihre Tauglichkeit überprüft worden waren. Innerhalb kurzer Zeit nahmen deshalb die Freifunker umfangreiche Veränderungen an der Implementierung vor.

3.2 Die OLSR-Implementierung von Olsr.org und Freifunk

Klassische Link-State-Protokolle verwenden für das Routen von Datenpaketen stets den kürzesten Pfad (Minimum Hop Count). Auch OLSR nach RFC 3626 geht diesen Weg. Dadurch werden potentiell Routen über schwache und instabile Funkstrecken bevorzugt, um die Anzahl der Sprünge über Zwischenstationen klein zu halten. Die Stabilität und der Datendurchsatz einer solchen Route ist nur in Ausnahmefällen brauchbar. Ein sinnvollerer Algorithmus zur Bestimmung der Routen wurde benötigt, der den besten Kompromiss aus Bandbreite und Routenlänge finden sollte. Diese Problematik führte zur Entwicklung des Linkquality-ETX-Algorithmus für OLSR bei Freifunk. Zwei Konzepte, die zur Optimierung des Link-State-Verfahrens in OLSR dienen sollten, wurden ebenfalls unmittelbar nach dem miserablen Resultat auf der „Wizards of OS" über Bord geworfen: Hysterese und Multipoint-Relais. Da man bei der Arbeit mit OLSR immer wieder auf diese Konzepte stößt, soll hier kurz darauf eingegangen werden.

3.2.1 Hysteresealgorithmus

Der Begriff *Hysterese* bedeutet so viel wie Beharrungsvermögen. Bei OLSR verbirgt sich dahinter ein Mechanismus, der entscheidet, ob benachbarte Meshknoten für das Routen von Paketen überhaupt in Betracht gezogen werden. Um allzu schwache Verbindungen zu vermeiden, werden benachbarte Router aus der Routing-Tabelle entfernt, wenn eine bestimmten Anzahl von erwarteten, aufeinanderfolgenden Hello-Paketen ausbleibt.

Für die Entscheidung, ob ein benachbarter Router verwendet wird, werden ein unterer und ein oberer Schwellwert definiert. Empfängt ein OLSR-Router beispielsweise von einem Nachbarn weniger als drei von zehn aufeinander folgenden Hello-Paketen, wird dieser aus der Routingtabelle gelöscht (unterer Schwellwert der Hysterese). Erst wenn beispielsweise sieben von zehn aufeinander folgenden Hello-Paketen wieder eingetroffen sind (oberer Schwellwert der Hysterese), wird der Nachbar wieder zur Routingtabelle hinzugefügt. Wurde ein benachbarter Router aus der Routingtabelle gelöscht und schafft dieser es nicht, mehr als sechs von zehn Hello-Nachrichten erfolgreich zu übertragen, sorgt die Hysterese dafür, dass er nicht mehr in die Routingtabelle aufgenommen wird.

Da Datenfunkstrecken immer wieder durch Störungen irgendwelcher Art einige Sekunden aussetzen können, werden durch den Hysteresealgorithmus benachbarte Router und die über sie erreichbaren Nodes immer wieder aus der Routingtabelle geworfen. Das ist verheerend für Anwendungen wie Webbrowser, Mailclients oder VoIP, die über das Mesh Daten transportieren wollen, da sie jedes Mal die Meldung vom Betriebssystem erhalten,

dass keine Verbindung möglich ist. Die Programme geben dann selbst eine Fehlermeldung aus und brechen die Verbindungsversuche ab, obwohl die Verbindung nur kurz aussetzt.

OLSR verhält sich dabei leider nicht opportunistisch – wenn es nur einen einzigen Nachbarn gibt, um ein entferntes Ziel zu erreichen, ist es sehr ungeschickt, diesen Router aus der Routingtabelle zu löschen. Besser ist es, wenn das Routingprotokoll erkennt, dass es nur eine – wenn auch schlechte – Verbindung hat, und diese so weit wie möglich nutzt. Der Hysteresealgorithmus tut das aber leider nicht.

Da OLSR bei der Routenfindung versucht, die Anzahl der Zwischenstationen zu reduzieren, wählt es bevorzugt solche Nachbarn als Relaisstationen, die über instabile Verbindungen erreicht werden. Diese fallen dann ständig wieder aus der Routingtabelle – einschließlich der anderen Meshknoten, die nur über diesen Nachbarn erreichbar waren. Die Folge sind in der Praxis Verbindungsabbrüche, fortlaufende Fehlermeldungen und immer wieder zusammenbrechende Routingtabellen. Der Hysteresealgorithmus von OLSR wird deshalb von Freifunk nicht verwendet.

3.2.2 Multipoint-Relais

Link-State-Protokolle übertragen sehr viele Protokolldaten ins Netz, damit jeder Netzwerkknoten sich zeitnah ein einigermaßen stimmiges Bild über den Zustand der Gesamttopologie machen kann. Um die Menge an einzelnen Protokolldatentransfers zu reduzieren, haben sich die Erfinder von OLSR mit den Multipoint-Relais eine Sparmaßnahme einfallen lassen: Nicht jeder Knotenpunkt im Mesh soll Topologiedaten weiterreichen, sondern nur jene Knotenpunkte, die für das Fluten des Gesamtnetzes unbedingt notwendig sind. Dazu wählt ein OLSR-Router nur jene direkten Nachbarn (Single Hop) aus, die theoretisch notwendig sind, um alle mehr als zwei Hops entfernte Nachbarn zu erreichen. Es wird also eine Untermenge der direkten Nachbarn berechnet. Nur diese beauftragt ein OLSR-Knoten damit, als Multipoint-Relais seine Topologieinformationen weiterzuleiten.

Fatalerweise wirft der OLSR-Hysteresemechanismus gerade diese Netzwerkknoten besonders häufig aus der Routingtabelle, denn um die Untermenge klein zu halten, werden diejenigen direkten Nachbarn bevorzugt, die relativ weit entfernt und deshalb unzuverlässig sind.

Nun muss das OLSR-Protokoll die Topologieinformationen aber trotzdem verbreiten, also andere Multipoint-Relais aushandeln, was Zeit kostet. Währenddessen bleiben Topologieinformationen aus, die Topologiedatenbanken der Nodes werden nicht rechtzeitig synchronisiert. Es kommt unter den Routern zu Unstimmigkeiten über den Pfad, auf dem die Datenpakete weitergereicht werden sollen. Die Folge sind Kreisrouten: Datenpakete irren herum, bis ihre Lebensdauer abgelaufen ist.

3.2.3 Olsr.org – Aus den Fehlern lernen

Das Wechselspiel scheinbar guter Ideen sorgt in der Praxis also für Chaos im Netzwerk. Paradoxerweise arbeitet ein einfaches Linkstate-Protokoll ohne Optimierungen besser als RFC 3626. Bis heute hält die INRIA aber in ihren weiteren IETF-Entwürfen zu OLSR an diesen Konzepten fest. Doch kaum jemand wird heute außerhalb der wissenschaftlichen Forschung OLSR nach RFC 3626 einsetzen, auch wenn das mit der Routingsoftware `olsrd` von Olsr.org weiterhin möglich ist.

Eigentlich sollte man den modifizierten Routingdaemon nicht mehr als OLSR bezeichnen, da er die wichtigsten Alleinstellungsmerkmale des Protokolls, Hysterese und Multipoint-Relais, nicht verwendet. Trotzdem wird sich die Bezeichnung auch weiterhin halten und unter Anwendern für Verwirrung sorgen.

Der OLSR-Daemon `olsrd` von Olsr.org ist lauffähig unter Linux, FreeBSD, NetBSD, OpenBSD, Mac OS X, Windows 98 SE, Windows ME, Windows NT, Windows 2000 und Windows XP. Nicht unterstützt werden Symbian und Windows CE beziehungsweise Windows Mobile.

Es ist ein Dienstprogramm, das üblicherweise im Hintergrund läuft und nur in der Liste der laufenden Prozesse in Erscheinung tritt. Üblicherweise wird `olsrd` von der Kommandozeile gestartet. Für Microsoft Windows gibt es zur Bedienung und Konfiguration die grafische Benutzeroberfläche Olsr-Switch. Bei Routern mit der Freifunk-Firmware kann `olsrd` von der Weboberfläche konfiguriert und beobachtet werden.

3.2.4 Funktionsweise des OLSR-Daemon

Beim Start des OLSR-Daemon beginnt dieser, über alle zugeordneten Netzwerkkarten in einem definierten Intervall an unmittelbare Nachbarn Hello-Nachrichten per Broadcast zu senden. Die Nachbarn schicken ihrerseits selbst Hello-Nachrichten, die vom Olsrd empfangen werden, wenn die physikalischen Bedingungen es zulassen. Dadurch lernt Olsrd, welche Nachbarn sich in unmittelbarer Funkreichweite befinden. Die Nachbarn erfahren ihrerseits von der Anwesenheit des neuen Nodes. In den Hello-Nachrichten teilen die Absender mit, in welchem Intervall sie Hello-Nachrichten verschicken.

Der Freifunk-Olsrd führt anhand dieser Informationen eine Statistik darüber, wie viele Hello-Nachrichten von jedem Nachbarn innerhalb einer bestimmten Zeit erwartet werden können und wie viele tatsächlich eingetroffen sind. Olsrd führt diese Statistik über eine definierte Anzahl von Paketen. Ein Meshknoten lernt also: Ich habe drei Nachbarn A, B, C. Von Nachbar A hätte ich innerhalb der letzten 500 Sekunden 100 Hellos empfangen müssen, aber nur 33 erhalten. Von Nachbar B bekam ich 90 Hellos von 100, von

Nachbar C 60 Hellos von 100. Das Ergebnis seiner Statistik teilt der Olsrd seinen Nachbarn mit, indem er diese Informationen stets aktuell in seine Hello-Pakete schreibt. Die Nachbarn tun ihrerseits dasselbe.

Angenommen, es existiert noch ein Nachbar D, von dem gelegentlich Hello-Pakete empfangen werden, aber dieser Nachbar D sieht unseren Olsrd nicht – dann tauchen wir in seiner Statistik, die er dem Netz in seinen Hellos mitteilt, nicht auf. Wir wissen also, dass dieser Knoten uns nicht sieht.

Auf diese Weise erfahren alle Nachbarn nicht nur, wie gut sie andere sehen, sondern auch, wie gut sie gesehen werden. Da der Olsrd nun weiß, wie häufig, statistisch gesehen, eine Übertragung von einem Nachbarn zu ihm klappt (Link-Qualität, abgekürzt LQ), beziehungsweise eine Übertragung von ihm zu dem Nachbarn (Nachbar-Link-Qualität, NLQ), kann Olsrd die Wahrscheinlichkeit ausrechnen, mit der ein Paket, welches eine Rundreise zum Nachbarn und wieder zurück antritt, wieder eintrifft.

Diese Wahrscheinlichkeit – wie viele Pakete müssen statistisch eine Rundreise antreten, damit ein Paket wieder erfolgreich zurückkommt – wird in Anlehnung an eine Studie am Massachusetts Institute of Technology (MIT) über eine sinnvolle Metrik zur Beurteilung von Routen für drahtlose Netzwerke *ETX* (*Expected Transmission Count*). Die Berechnung ist einfach. Es ist der Kehrwert der Round-Trip-Wahrscheinlichkeit:

$$ETX = 1/(LQ * NLQ)$$

Im Idealfall hat eine Funkstrecke einen ETX-Wert von 1,00. Das heißt, jedes ausgesendete IP-Paket kommt auf seiner Rundreise auch wieder zurück. Ein ETX-Wert von 4 bedeutet, dass vier Pakete verschickt werden müssen, damit ein Paket wieder zurückkommt. Bei einem unbrauchbaren Funklink, der nur in einer Richtung funktioniert, ist der ETX-Wert unendlich.

Mit den ETX-Werten kann Olsrd bei Routingentscheidungen durch einfaches Addieren der ETX-Metriken die günstigste Route bestimmen. Das Verfahren berücksichtigt, dass jede Zwischenstation auf einer Route Bandbreite und Geschwindigkeit kostet. Der Olsrd muss abwägen zwischen Routen mit vielen Hops, auf der jede Funkstrecke mit gutem Durchsatz arbeitet, und Routen über einen kürzeren Pfad, wo auf überdehnten Linkstrecken nur wenige Datenpakete durchkommen. Der beste Pfad ist der Mittelweg zwischen beiden Extremen, da sich bei jeder Übertragung über ein Funkrelais die Bandbreite deutlich reduziert, wenn ein Meshrouter nur mit einer WLAN-Karte arbeitet (siehe Abschnitt 2.1.1). Andererseits sind überdehnte Funkstrecken nutzlos, selbst wenn noch eine Verbindung nachweisbar ist.

3.2.5 Topologieinformationen und Dijkstra-Algorithmus

Der Olsrd hat nun über Hello-Broadcasts von der Existenz direkter Nachbarn erfahren und kennt auch die Qualität der Funkverbindungen (LQ/

NLQ/ETX-Werte). Diese Topologieinformationen werden in einem IP-Paket verschickt, mit dem in einem bestimmten Intervall das ganze Meshnetz geflutet wird. Bei Olsrd heißt ein Datenpaket mit Topologieinformationen *TC-Message* (Topology Control Message). Anders als die Hello-Nachrichten, die nur lokal übertragen werden, werden Topologienachrichten von allen anderen Knoten im Netz sofort weitergereicht. Jeder Knoten merkt sich den Inhalt der herumschwirrenden Topologienachrichten und trägt sie zu einer eigenen großen Datenbank – einer „Landkarte" des Meshnetzes – zusammen. Auf diese Graphen wird dann der Link-State-Algorithmus von Dijkstra angewendet. Freifunk-Olsrd sucht anders als viele andere Link-State-Protokolle nicht nach der geringsten Anzahl von Hops, sondern nach dem günstigsten (kleinsten) ETX-Gesamtwert für jede Route.

Topologienachrichten und Protokolloverhead

Eine Überschlagsrechnung zeigt, dass ein klassisches Link-State-Verfahren für ein drahtloses Computernetzwerk wegen der CPU-Last und der vom Protokoll erzeugten Datenmenge mit wachsender Knotenanzahl schnell an seine Grenzen stößt. Wenn jeder Knoten alle fünf Sekunden eine neue Topologienachricht losschickt und damit eine ganze Lawine von Datentransfers im gesamten Netz auslöst, wird das schon bei 200 Knoten problematisch, da die Anzahl der Nachrichten quadratisch von der Knotenzahl abhängt:

$$\text{TC-Nachrichten/s} = \text{Knoten}^2 / \text{TC-Intervall}$$

Statistisch wird so alle 25 Millisekunden ein anderer Netzteilnehmer eine neue Topologienachricht auf die Reise schicken und kann damit theoretisch eine Lawine von bis zu 199 Wiederholungen lostreten. Jede eintreffende Topologienachricht mit neuen Informationen verursacht dabei ein erneutes Durchrechnen des gesamten Netzwerkgraphen in jeder Maschine. Es können theoretisch bei 200 Knoten im Mesh und einem TC-Intervall von fünf Sekunden bis zu 8000 Topologienachrichtenübertragungen pro Sekunde anfallen. Bei 400 Knoten könnten es bereits bis zu 32 000 TC-Broadcasts pro Sekunde sein.

In der Praxis ist der Protokolloverhead (die durch OLSR produzierte Datenmenge) wesentlich kleiner als in der Theorie. Da alle Protokollnachrichten verbindungslos als Broadcast versendet werden, werden die Daten von den Empfängern nicht erneut angefordert, wenn sie wegen Kollisionen oder Interferenzen auf der Funkstrecke beschädigt werden. Beschädigte Pakete werden einfach verworfen und sind verloren. Bei jedem Hop von Node zu Node tritt deshalb ein nicht unerheblicher Schwund auf. Nach ein paar Hops ist die Wahrscheinlichkeit einer erfolgreichen Übertragung über eine Kette von Broadcasts nur noch gering und trägt erheblich dazu bei, dass der Protokolloverhead nicht unmittelbar ins Uferlose wächst.

Um Bandbreite zu sparen, werden TC-Nachrichten von OLSR nicht als einzelne kleine Pakete verschickt, sondern mit anderen TC-Nachrichten und weiteren Protokollinformationen zu größeren IP-Paketen gebündelt, da mit jedem Paket auch noch weitere Informationen wie IP-Header, UDP-Header, MAC-Adresse, Broadcast-Adresse als Overhead mit übertragen werden müssen. Durch größere Pakete reduziert sich der IP-Overhead gegenüber den zu übertragenden Protokolldaten. Trotzdem ist die von OLSR erzeugte Datenmenge groß und verbraucht einen nicht unbeträchtlichen Teil der Bandbreite, der nicht mehr für die Nutzlast des Netzwerks zur Verfügung steht.

Kurze TC-Intervalle scheinen sich daher per se in größeren Meshnetzwerken zu verbieten. Andererseits ist aber ein häufiger und redundanter Abgleich der Datenbank in den Knoten unbedingt erforderlich, da es sonst beim Routen zu Unstimmigkeiten über den Verlauf des günstigsten Pfads kommt (Routing-Loops). In IP-Paketen befindet sich ein Zähler (Time-To-Live, TTL), dessen Wert bei jedem weiteren Transportschritt um 1 heruntergezählt wird. Wird der Wert der TTL Null, wird das Paket verworfen und eine Meldung an den Absender geschickt. Auf diese Weise wird verhindert, dass verirrte Pakete bis in alle Ewigkeit im Netz herumsausen.

Das kann man sehr gut in einem Meshnetzwerk mit `ping` beobachten, indem man eine entfernte Maschine anpingt. Taucht die Meldung `Time to live exceeded` auf, ist das Ping-Paket so lange im Kreis herumgelaufen, bis der darin enthaltene TTL-Zähler bei Null angekommen ist.

Der Fisheye-Algorithmus

Der Multipoint-Mechanismus in RFC 3626 mildert kaum das potenzierte Wachstum der Topologieinformationen in größeren Meshnetzwerken. Selbst wenn nur jeder zweite oder dritte Knotenpunkt an der Weiterleitung von Topologienachrichten beteiligt ist, kann das Wachstum der TC-Pakete bei steigender Knotenanzahl mit Multipoint-Relais kaum wirkungsvoll begrenzt werden. Auch zu dieser Problematik ist den Freifunkern etwas eingefallen: der Fisheye-Algorithmus. Das Prinzip ist ganz einfach: Die TTL (auch TC-Messages enthalten eine TTL) wird bei den meisten TC-Broadcasts auf sehr kleine Werte zwischen 1 und 3 gesetzt. Auf diese Weise wird erreicht, dass diese TC-Nachrichten nur ein bis drei Hops weit durch das Mesh reisen, bis die TTL abgelaufen ist und die Nachricht verworfen wird.

Nur selten wird eine TC-Message mit einer großen TTL auf die Reise geschickt, die sich über das gesamte Netzwerk ausbreiten kann – wenn sie nicht unterwegs verloren geht. Topologienachrichten mit kleinen TTL-Werten werden sehr oft geschickt, sogar öfter als Hello-Nachrichten. Dadurch wird erreicht, dass Änderungen der ETX-Werte durch gerade eintreffende oder verlorengegangene Hellos mit einer gewissen Redundanz den lokalen Nachbarn mitgeteilt werden. Auf diese Weise können Routing-Loops weitgehend verhindert werden.

Kreisrouten treten nur zwischen Netzwerkknoten auf, die nur wenige Hops voneinander entfernt liegen. Es ist unwahrscheinlich, dass auf einer Kreisroute ein IP-Paket erst drei Hops nach Norden, drei Hops nach Osten, drei Hops nach Süden und dann wieder drei Hops nach Westen reist. Meistens sind insgesamt nur zwei oder drei Knoten an einer Kreisroute beteiligt. Entscheidend für einen reibungslosen Transport der Nutzlast ist, dass benachbarte Knoten sich untereinander über ihr Bild der Netztopologie einig sind. Solange auf einer Route die Richtung stimmt und sich die benachbarten Router über den lokalen Verlauf einig sind, bewegt sich das IP-Paket auf sein Ziel zu. Dabei nähert es sich Nodes, die umso besser über die Netztopologie Bescheid wissen, je näher sie dem Zielpunkt liegen.

Wenn man ein Postpaket in ein Dorf in Indien schickt, muss man nicht wissen, welchen Weg der indische Paketbote mit seinem Fahrrad einschlägt, oder welche Farbe sein Fahrrad hat. Der Paketbote vor Ort kennt den Weg viel besser als der Absender im fernen Europa. Wir müssen nur wissen, wo das nächstgelegene Postamt liegt, das Sendungen nach Asien annimmt. Ebenso verhält es sich im Meshnetz. Ein Node muss nur wissen, welchem Nachbarn er zweckmäßigerweise das IP-Paket zum Weitertransport anvertraut. Die Entscheidung, was dieser Nachbarknoten mit dem IP-Paket anstellt, trifft dieser selbstständig und unabhängig von der Sicht der anderen Knoten.

Topologieinformationen von weit entfernten Knoten sind möglicherweise schon sehr alt, da auf dem Weg durch das Meshnetz viele TC-Nachrichten verlorengehen. Aktuellere Informationen, die eine falsche Sicht über das andere Ende des Netzwerks korrigieren können, kommen nur selten oder verspätet an. Das macht aber nichts, da ein Node nicht vorgeben muss, welchen genauen Weg ein Paket am anderen Ende des Netzwerkes zu nehmen hat. Wegen der hohen Dynamik des Netzwerks könnte der Router das auch gar nicht.

Es genügt also, wenn ein Knoten nur eine unscharfe Sicht von der Topologie des Netzes in der Entfernung hat. Er muss über die Topologie in der Entfernung nur wissen, wer in welcher Richtung vorhanden ist, aber nicht wo genau. Daher kann sich OLSR das permanente Durchrechnen des kompletten Graphen wegen jeder einzelnen TC-Message aus der Ferne sparen. Deshalb genügt es, wenn nur selten TC-Messages aus der Ferne auftauchen. Über lokale Veränderungen sollte ein Meshknoten umso genauer Bescheid wissen.

Der Fisheye-Mechanismus enthält daher zwei Komponenten: Zum einen werden TC-Nachrichten bei ihrer Ausbreitung durch die kleinen TTL-Werte überwiegend auf einen lokalen Rahmen beschränkt. Zum anderen lösen nur TC-Nachrichten von Nodes in der Nähe eine Neuberechnung des Graphen aus. Im aktuellen Olsrd wird nur jede 13. TC-Message mit einer TTL für die „weite Welt" versehen. Bei einem Emissionsintervall von fünf Sekunden wird nur alle 65 Sekunden das ganze Netz von einem Node geflu-

tet. Damit schrumpfen Netzwerklast und Rechenlast ganz erheblich, ohne die Stabilität des Netzwerks zu gefährden. Bei einer Netzwerkgröße von 500 Knoten können Nodes die TC-Messages alle zwei bis drei Sekunden auf die Reise schicken.

Auch der Rechenaufwand zur Neuberechnung des Netzwerkgraphen steigt mit der Anzahl der Knoten potenziert an. Bei 500 Knoten nimmt ein einmaliges Durchrechnen der Topologiedatenbank bei der geringen Rechenleistung eines SoHO-Routers bereits Sekunden in Anspruch. Das Durchrechnen des Graphen beim Eintreffen jeder neuen TC-Message ist auf einem kleinen Computer dann nicht mehr möglich.

Im Freifunk-Olsrd kann ein festes Intervall eingegeben werden, das angibt, wie oft die Gesamttopologie durchgerechnet wird. Das ist leider keine saubere Verfahrensweise, da damit wieder Probleme wie Routingloops auftreten können. An dieser Stelle kann nur noch ein verändertes Dijkstra-Verfahren helfen, welches auch die Netzwerktopologie nur noch bis zu einer bestimmten Anzahl Hops aus lokaler Sicht durchrechnet. Dieses Verfahren ist allerdings im Moment noch nicht implementiert und wartet darauf programmiert zu werden.

3.3 Praxis mit Olsrd

Olsrd ist selbstverständlich in der Freifunk-Firmware enthalten. Für OpenWrt, Windows und Mac OS X gibt es Installationspakete. Doch die meisten Funktionen bietet die Linux-Variante von Olsrd. Der Funktionsumfang des Olsrd kann über Bibliotheken, die zur Laufzeit in das Programm geladen werden (Plugins) erweitert werden. Da Olsrd überwiegend unter Linux entwickelt und eingesetzt wird, funktionieren nur unter Linux alle Programmfunktionen ohne Einschränkungen.

3.3.1 Installation unter Linux, Mac OS X und *BSD

Soll Olsrd auf einem Laptop oder Desktop-PC mit Linux oder einer Unix-Variante wie FreeBSD installiert werden, ist es empfehlenswert, sich die aktuellen Programmsourcen von der Webseite olsr.org herunterzuladen und selbst zu kompilieren. Debian GNU/Linux oder Ubuntu haben zwar Olsr-Pakete, die über das Paketmanagementsystem installiert werden können, diese sind jedoch üblicherweise veraltet.

Die Entwickler von Olsr.org haben bislang nur selten eine neue offizielle Veröffentlichung der Olsr-Software herausgegeben. Die jeweils letzte offizielle Programmversion enthält mitunter längere Zeit Fehler, die in der Entwicklerversion bereits behoben sind. Sollte die letzte offizielle Programmveröffentlichung schon einige Zeit zurück liegen, ist es wahrscheinlich bes-

ser, eine aktuelle Entwicklerversion zu verwenden. Der aktuelle Entwicklungsstand des Olsrd ist leider nicht als komprimiertes Dateiarchiv zum Download erhältlich.

Installation über die Programmsoureen unter Linux und *BSD

Die Quelltexte der Software sollten deshalb aus dem Versionsmanagementsystem von Olsr.org heruntergeladen werden. Dazu muss auf dem lokalen Rechner die passende Software für das Versionsmanagementsystem CVS (Concurrent Versions System) installiert sein.

Dazu wird in einem Kommandozeileninterface der folgende Befehl ausgeführt, um sich mit dem Server zu verbinden.

```
cvs -d:pserver:anonymous@olsrd.cvs.sourceforge.net:/cvsroot/olsrd login
```

Wenn der Server nach einem Passwort fragt, genügt es, die Eingabetaste zu drücken. Der nun folgende Befehl startet den Download. Genauer: Das lokale Verzeichnis wird mit dem Inhalt des Servers synchronisiert.

```
cvs -z3 -d:pserver:anonymous@olsrd.cvs.sourceforge.net:/cvsroot/olsrd \
  co olsrd-current
```

CVS legt nun in dem Verzeichnis, in dem der Befehl gestartet wurde, ein neues Unterverzeichnis namens `olsrd-current` an. Soll später auf einen neueren Stand der Software aktualisiert werden, überträgt CVS nur noch die Veränderungen, wenn es wieder im gleichen Verzeichnis gestartet wird wie zuvor.

Es ist nicht garantiert, dass der aktuelle Stand der Entwicklung aus dem CVS funktioniert. Meistens ist das jedoch der Fall. Sollte der Inhalt des CVS einmal nicht funktionieren, kann man auf die nächste Version warten, die innerhalb kurzer Zeit verfügbar sein dürfte.

Ist auf der lokalen Maschine eine funktionsfähige Umgebung zum Kompilieren von C-Programmen vorhanden, ist die Installation relativ einfach, da die Software keine besonderen Paketabhängigkeiten aufweist. Ist man auf der Kommandozeile in das Verzeichnis mit den Olsr-Quelltexten gewechselt, genügen die beiden Befehle `make` und `make install`. Für letzteren benötigt man Administratorrechte.

Für Olsrd existieren einige interessante Erweiterungen (Plugins), auf die man nicht verzichten sollte. Die Quelltexte dafür sind zwar im Paket dabei, aber die Plugins müssen getrennt kompiliert und installiert werden. Dazu in das Unterverzeichnis `lib` im Sourceverzeichnis `olsrd-current` mit dem Befehl `cd lib` wechseln, dann die Befehle `make` und `make install` noch einmal ausführen.

Installation auf Ma OS X

Auf der Downloadseite von Olsr.org befindet sich ein Link zu vorkompilierten Olsr-Installationspaketen für Mac OS X, die von Maximilian Thumfart zusammengestellt werden[1]. Es werden sowohl Intel-Macs und PPC-Macs unterstützt. Die Mac-Version des Olsrd ist in der Regel auf einem aktuellen Stand. Größere Veränderungen im Routingdaemon fließen innerhalb kurzer Zeit auch in die Mac-Version ein. Natürlich können ambitionierte Mac-Anwender auch selbst kompilieren.

Die Installationspakete von Maximilian Thumfart enthalten eine zu Freifunk kompatible Konfigurationsdatei für den OLSR-Daemon. Nach der Installation kann der Olsrd in einem Terminalfenster über den Befehl sudo `olsrd` gestartet werden.

3.3.2 Installation unter Windows

Eigentlich ist Olsrd ein reines Kommandozeilenprogramm. Jedoch existiert mit dem *Olsr-Switch* exklusiv für Windows ein grafisches Frontend, das Bedienung und Konfiguration mit der Maus ermöglicht.

Auf Olsr.org befindet sich die Datei `olsr-0.4.10-setup.exe` unter der Rubrik **Downloads**. Ein komfortables Installationsprogramm führt per Mausklick die notwendigen Schritte durch und fordert anschließend zum einmaligen Neustart auf. Nach der Installation befinden sich die Programme im Verzeichnis `c:\Programme\olsr.org`. Leider wurde bis zur Drucklegung des Buches diese Version des Olsrd für Windows nicht mehr weiterentwickelt. Das graphische Frontend Olsr-Switch gibt sich in der Menüleiste fälschlicherweise als Programmversion 0.4.9 aus, der Olsr-Daemon entspricht aber der Version 0.4.10. Auch diese Version ist leider veraltet und enthält schon seit längerer Zeit bekannte Fehler. Auch der Fischaugen-Algorithmus ist nicht vorhanden.

Die Rechenlast ist deutlich höher als bei aktuellen Versionen von Olsrd. Da ein PC über wesentlich mehr Rechenleistung verfügt als ein SoHO-Router, ist das jedoch keine gravierende Einschränkung. Auf sehr langsamen Windows-Rechnern (Pentium II oder III) sollte jedoch auf den Komfort des grafischen Frontends nach der Konfiguration verzichtet werden, da das Frontend im Betrieb des Olsrd einiges an Rechenleistung frisst. Besser ist es, den Olsrd auf der Kommandozeile zu starten.

Die gravierendste Schwachstelle der veralteten Programmversion ist aber, dass Olsrd abstürzt, wenn Routen mit mehr als 31 Hops im Netzwerk vorkommen. Dieses Problem tritt natürlich nur in sehr großen Meshnetzwerken auf. Im Freifunk-Mesh in Berlin führt dieser Fehler zum häufigen Abstürzen des Programms.

[1] `http://www.fernsehsofa.de/olsr/`

Vor dem ersten Start des Olsrd muss dieser mit dem Olsr-Switch konfiguriert werden. Dabei muss unbedingt die Checkbox **Offer Internet Gateway** deaktiviert werden, wenn nicht wirklich ein Internet-Gateway über diesen PC angeboten werden soll. Andernfalls wird der Windows-PC für den Internetverkehr benachbarter Meshteilnehmer und zum Schwarzen Loch: Es wird ein Internetzugang offeriert, in dem die Internetanfragen anderer Netznutzer verschwinden.

Wenn der PC tatsächlich als Internet-Gateway dienen soll, ist es nicht ratsam, das mit der Windows-Version zu realisieren. Für Olsrd unter Linux gibt es ein dynamisches Gateway-Plugin. Dieses Plugin prüft, ob der Internetzugang wirklich funktioniert, und stellt das Annoncieren eines Internet-Gateways bei einer Fehlfunktion ein. Die Windows-Version überprüft das Funktionieren des Internetzugangs nicht.

Die im Screenshot 3.1 gezeigte Konfiguration kann als Ausgangsbasis übernommen werden. Das verwendete Interface kann von der Abbildung abweichen.

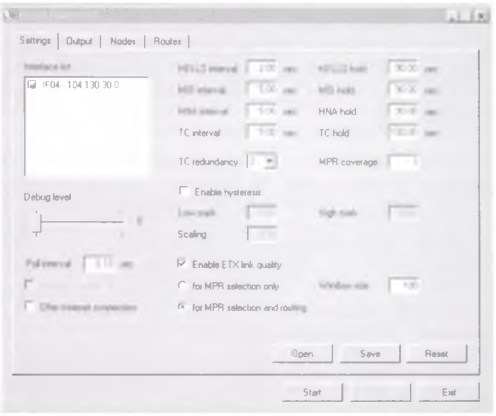


Abbildung 3.1: Grundkonfiguration mit Olsr-Switch

Sind alle Einstellungen vorgenommen, sollte die Konfiguration über die Schaltfläche **Save** gespeichert werden, bevor zum ersten Mal gestartet wird. Die Default-Konfiguration kann man dabei getrost überschreiben. Eine aktive Firewall meldet beim ersten Start, dass Olsr-Switch den Port 698 verwenden möchte. Das muss natürlich zugelassen werden. Über die Reiter **Output**, **Nodes** und **Routes** kann die Funktion beobachtet werden. Tauchen keine Nodes und Routes auf, kann ein Blick unter dem Menü **Output** helfen, wenn das Debug-Level größer oder gleich 1 gesetzt wurde. Befindet

man sich in einem größeren Mesh, kann man den Output kaum lesen, da bei der Flut von Meldungen die Ausgabe nicht mehr stillsteht. Das Durchrauschen der Anzeige ist ein gutes Zeichen, so lange da nicht seitenweise die Meldung erscheint: `Received Packet from Non-Sym-Neighbor`. Dann stimmt meistens etwas mit der WLAN-Karte nicht oder die anderen Meshknoten empfangen das lokale Gerät nicht.

Besser und ressourcenschonender kann der Status des Olsrd durch das OLSR-Plugin Httpinfo in einem Webbrowser beobachtet werden. Dazu wird mit einem Editor die Konfigurationsdatei um die notwendigen Eintragungen für die Nutzung des Plugins erweitert. Informationen zur Einrichtung von Httpinfo finden sich in Abschnitt 3.4.2.

Einmal eingerichtet, kann Olsrd auf der Kommandozeile aus dem Verzeichnis gestartet werden, in dem der Daemon installiert wurde. Die zu verwendende Konfigurationsdatei muss über die Option `-f` angegeben werden.

```
olsrd.exe -f Default.olsr
```

Da der OLSR-Daemon sich vollständig über die Konfigurationsdatei anpassen lässt, ist die Angabe weiterer Startoptionen nicht notwendig. Lediglich wenn etwas nicht funktioniert wie gewünscht, kann das Debuglevel beim Programmaufruf übergeben werden. Die Konfigurationsdatei muss stets zuerst angegeben werden, sonst bricht der Daemon den Start mit einer Fehlermeldung ab.

```
olsrd.exe -f Default.olsr -d 1
```

Eben weil Olsr-Switch es Anfängern besonders einfach macht, durch einen falschen Mausklick oder eine falsch konfigurierte Firewall das Netz empfindlich zu stören, ist die Unterstützung für Windows bei den Freifunkern nicht besonders beliebt und hinkt hinterher. Das soll nicht heißen, dass `olsrd-0.4.10` unter Windows nicht zufriedenstellend funktioniert. Technisch ist es allerdings auf dem Stand von Ende 2005 und eignet sich deshalb nur für Meshnetze, die nicht allzu groß sind.

Klappt die Verbindung zum Mesh nicht, sollte man die WLAN-Einstellungen, die Firewall, IP-Adresse, Netzwerkmaske und Broadcastadresse überprüfen. Das richtige Interface muss im Olsr-Switch ausgewählt werden. Oft verursachen die Treiber oder Firmware der WLAN-Karten Probleme. Eine solide Funktionsweise des Ad-Hoc-Modus ist bei WLAN-Karten für PCs leider eher die Ausnahme. Näheres zu dieser Problemstellung ist in Abschnitt 5.4 zu lesen.

3.3.3 Konfigurationsdatei des Olsrd

Vor dem ersten Start von Olsrd muss die Konfigurationsdatei editiert werden. Normalerweise werden dem Daemon keine Optionen beim Start über-

geben. Das macht bei der Vielzahl von möglichen Optionen und Einstellmöglichkeiten keinen Sinn. Lediglich die Angabe eines Debuglevels größer oder gleich 1 ist zur Fehlersuche sinnvoll.

Das Laufzeitverhalten des Olsr-Routingdaemons wird unter allen Betriebssystemen von der Konfigurationsdatei `olsrd.conf` gesteuert, diese wird normalerweise im Verzeichnis `/etc` installiert. Unter Windows sucht Olsrd im Windows-Systemordner nach der Datei `c:\windows\olsrd.conf` oder unter Windows NT `c:\WINNT\olsrd.conf`. Leider wird grundsätzlich immer noch RFC 3626 als Grundeinstellung installiert. Im Quelltextverzeichnis des Olsrd findet man im Unterverzeichnis `files` die Datei `olsrd.conf.default.lq-fisheye`, die manuell nach `/etc` kopiert und in `olsrd.conf` umbenannt werden muss.

Alle kritischen Einstellungen, welche das Verhalten von OLSR vorgeben, sind in dieser Konfigurationsdatei bereits vorgenommen. Die Konfigurationsdatei sollte in allen Routern in einem Mesh bis auf kleinere Anpassungen und Sonderfunktionen gleich sein. Im einfachsten Fall müssen lediglich die Netzwerkinterfaces eingetragen werden, auf denen der Olsr-Daemon routen soll. Durch die Verwendung einer Standardkonfiguration erschließt sich jedoch nicht das volle Potenzial des Olsrd. So ist zum Verwenden der Plugins etwas Handarbeit notwendig. Leider ist die `olsrd.conf` alles andere als selbsterklärend. Wenn man sich nicht auf eine fertige Meshlösung wie die Freifunk-Firmware verlassen will oder eigene Wünsche und Vorstellungen hat, muss man sich in die Untiefen der Konfigurationsdatei begeben.

Die im Folgenden vorgetellte Konfigurationsdatei enthält unkommentiert alle wichtigen Konfigurationsparamenter. Die einzelnen Komponenten werden im anschließend folgenden Abschnitt detailliert erklärt. Diese Datei ist eine typische Beispielkonfiguration für ein Meshnetzwerk aus überwiegend stationären Knoten, wie sie von Community-Netzwerken wie Freifunk verwendet wird. Eine ausführliche, in deutscher Sprache kommentierte Konfigurationsdatei finden Sie im Internet[2].

Das Editieren von Konfigurationsdateien darf nur mit einem Texteditor wie Notepad unter Windows oder `nano` oder `vi` geschehen, da Textverarbeitungsprogramme unerwünschte Formatierungszeichen hinzufügen, welche die Datei für den Daemon unlesbar machen. Unter Windows müssen die Konfigurationsdateien mit Notepad im Format UTF-8 gespeichert werden, sonst tritt der gleiche Effekt auf.

Wie unter Unix/Linux in vielen Konfigurationsdateien üblich, signalisiert ein Rautenzeichen am Anfang der Zeile einen Kommentar.

[2] `http://downloads.open-mesh.net/misc/olsr/olsr.conf.example-german-comments`

```
# olsr.org OLSR Daemon Konfigurationsdatei

DebugLevel    1
LinkQualityFishEye 1
LinkQualityDijkstraLimit 3 6.0
IpVersion       4
ClearScreen     yes

Hna4
{
    10.193.198.0 255.255.255.0
    10.140.140.254 255.255.255.255
}

Hna6
{
        #   Internet Gateway:
        #   ::              0
        #   Weitere Einträge:
        #   fec0:2200:106:: 48
}


AllowNoInt      yes
TosValue       16
Willingness     4

IpcConnect
{
    MaxConnections  1
}

UseHysteresis   no
LinkQualityLevel        2
LinkQualityWinSize      100
Pollrate        0.05
TcRedundancy    2
MprCoverage     7

LoadPlugin "olsrd_dyn_gw.so.0.4"
{
    PlParam     "Interval"   "40"
    PlParam     "Ping"       "141.1.1.1"
}

Interface "eth0" "ath0" "wlan0" "eth2"
{
    # Ip4Broadcast              255.255.255.255
    # Ip6AddrType               site-local
    # Ip6MulticastSite          ff05::11
    # Ip6MulticastGlobal        ff0e::1
    HelloInterval    2.0
```

```
    HelloValidityTime   200.0
    TcInterval        0.5
    TcValidityTime      200.0
    MidInterval 5.0
    MidValidityTime     200.0
    HnaInterval 5.0
    HnaValidityTime     200.0
    # LinkQualityMult 192.168.0.1 0.5
    # LinkQualityMult default 0.7
    # LinkQualityMult 10.143.143.143 1.0
    # LinkQualityMult 10.143.143.144 1.0
}

Interface "tun0"
{
    HelloInterval           50.0
    HelloValidityTime   2000.0
    TcInterval               50.0
    TcValidityTime            2000.0
    MidInterval         50.0
    MidValidityTime     1000.0
    HnaInterval         50.0
    HnaValidityTime     1000.0
}
```

`Debug level`
: Dieser Wert (von 0 bis 9) gibt an, welche Informationen der Daemon auf das Terminal ausgibt. Bei Null gibt er keine Informationen aus und geht sofort in den Hintergrund. In einem großen Netzwerk werden bei Werten größer Null so viele Informationen ausgegeben, dass die Ausgabe einfach durchrauscht. Man sollte sie dann über eine Pipe in eine Datei umleiten, um sie sich in Ruhe ansehen zu können.

  Noch besser ist es, das Olsr-Plugin Httpinfo zu laden und sich die Ausgaben von Olsrd bequem in einem Webbrowser anzeigen zu lassen. Debugging braucht eine beträchtliche Menge Rechenleistung, wenn das Netz einigermaßen groß ist. Für kleine Embedded-Systeme sollte der Wert auf 0 gesetzt werden.

`LinkQualityFishEye`
: Soll der Fischaugenmechanismus verwendet werden? 0 = aus, 1 = an.

`LinkQualityDijkstraLimit`
: Diese Einstellung verhindert, dass jede eintreffende neue Topologienachricht ein erneutes Durchrechnen der gesamten Netzwerkdatenbank auslöst. Das ist nicht nötig, wenn einige Sprünge weiter eine Veränderung auftritt. Allerdings sollten Veränderungen in der Topologie bei benachbarten Knoten unmittelbar berücksichtigt werden.

Hier müssen zwei Werte eingegeben werden. Der erste Wert - eine Ganzzahl - gibt die Anzahl der Sprünge an. Ein Wert von 2 bedeutet, dass nur neue Topologieinformationen, die von direkten Nachbarn (Single-Hop) oder Nachbarn zweiter Ordnung (Two-Hop) stammen, ein Durchrechnen der Routingtabelle auslösen. Wird der Wert auf Null gesetzt, löst keine eintreffende Topologienachricht eine Neuberechnung aus. Der zweite Wert, eine Zahl mit einer Kommastelle, gibt an, in welchem Intervall die Routingtabelle neu berechnet wird, wenn der erste Wert für den Hopcount auf Null gesetzt ist.

`IpVersion`
Version des Internetprotokolls. 4 = IPv4, 6 = IPv6.

`ClearScreen`
Soll die Anzeige im Terminal jedes Mal neu aufgebaut werden, wenn eine Zustandsänderung eintritt? `yes/no`

`Hna4`
HNA-IPv4-Routen (Host Network Announcement): Über eine HNA-Nachricht kann ein OLSR-Knoten mitteilen, dass er ein Gateway zu anderen Subnetzwerken oder Hosts ist. Auf diese Weise kann man z. B. Geräte ohne OLSR (VoIP-Telefone, PDAs) in das Meshnetz einbinden, ohne NAT zu verwenden. In den Geräten ohne Olsrd wird dann dieser OLSR-Router, der den HNA ankündigt, als Gateway angegeben (entweder von Hand oder über DHCP).

Dabei sollten die Administratoren der Meshknoten sich unbedingt darüber verständigen, wer welche Subnetzwerkadressen benutzt beziehungsweise ankündigt. Das Olsr-Plugin Httpinfo listet alle HNA-Ankündigungen im Mesh unter der URL `http://localhost:8080/nodes` auf. Ein prüfender Blick in die bereits existierenden HNAs kann nicht schaden, um Kollisionen zu vermeiden, bevor ein neues HNA eingerichtet wird. Ein beliebtes Problem ist, dass ein Meshrouter das gleiche Subnetzwerk per HNA ankündigt, das auch von einem anderen Olsr-Router im Mesh auf seinem Ethernetport verwendet wird. Vor allem die bekannten Klasse-C-Netzwerke 192.168.0.0 und 192.168.1.0 sollte man unbedingt vermeiden. Das kann nicht gut gehen!

Stattdessen unbedingt Subnetzadressen lokal verwenden beziehungsweise ankündigen, die nicht von den OLSR-Routern von Lieschen Müller und Otto Normalverbraucher benutzt werden. Verwenden Lieschen und Otto am Ethernetport ihrer OLSR-Router die Subnetzadresse 192.168.1.0/24, kann Lieschen von ihrem PC nicht mehr auf ihren Router zugreifen, wenn Otto eine HNA zu „seinem" Subnetz 192.168.1.0/24 ankündigt - obwohl sie direkt per Kabel mit ihrem Router verbunden ist. Lieschens OLSR-Router trägt wegen Ottos HNA einen benachbarten OLSR-Router als Gateway zu Ottos kollidierendem Sub-

netz ein. Jetzt kann Lieschen von ihrem PC über LAN nicht mehr ihren Router erreichen, da dieser seine IP-Pakete für Lieschens PC über das Mesh an Ottos Router schickt.

Für das Ankündigen eines Internet-Gateways (0.0.0.0/0) sollte immer das dynamische Gateway-Plugin verwendet werden. Sonst hat das Mesh bei einem Ausfall des Internetanschlusses immer wieder schwarze Löcher, in dem Pakete zum Internet verschwinden. Schlägt man diese Warnung in den Wind, stehen bald viele ärgerliche Netzteilnehmer vor der Tür und beschweren sich.

Mehrere Einträge sind möglich, einer Netzadresse folgt immer eine Netzmaske. Der erste Eintrag im Listing routet zu einem Klasse-C-Netz, der zweite zu einem einzelnen Node.

`Hna6`
: HNA-IPv6-Routen.

`AllowNoInt`
: Soll der Olsrd auch dann weiterlaufen, wenn keine Netzwerkkarte zur Verfügung steht? (`yes` oder `no`) Das kann für PCMCIA/USB-Geräte oder andere Hotplug-fähige Hardware hilfreich sein.

`TosValue`
: TOS (Type of Service) für den IP-Header der Protokollpakete. Der Standardwert ist 16.

`Willingness`
: Die fest eingestellte Bereitschaft, für andere Knoten Pakete weiterzurouten (`0-7`). Falls nicht angegeben, wird der Wert bei Geräten mit Stromsparmechanismen (ACPI, APM) dynamisch dem Ladezustand/Stromsparmodus angepasst, wenn diese Informationen verfügbar sind.

`IpcConnect`
: Diese Option erlaubt es einem Olsr-Frontend wie Olsr-Switch auf den Daemon zuzugreifen. `MaxConnections` gibt die Anzahl der erlaubten Verbindungen an.

`UseHysteresis`
: Soll die Hysteresefunktion verwendet werden? (`yes` oder `no`) Die Antwort heißt: Nein. Hysterese ist nicht kompatibel zu LQ-ETX. Vorsicht: Hysterese wird grundsätzlich verwendet, wenn dieser Eintrag auskommentiert ist!

`LinkQualityLevel`
: Link Quality Extension (LQ-ETX) wird bei einem Wert von 0 nicht verwendet, bei 1 verwendet, um Multipoint-Relais zu bestimmen, beim

Wert 2 bestimmt sie Multipoint-Relais und Routen. Ist diese Einstellung nicht gesetzt, wird LQ-ETX nicht verwendet. Das ist keine gute Idee. Hier muss 2 stehen.

`LinkQualityWinSize`
: Link-Qualität-Fenstergröße zur Berechnung von ETX - Anzahl der Hello-Pakete für die ETX/LQ-Statistik. Nicht zu klein wählen, solange nicht hohe Mobilität der Netzwerkknoten berücksichtigt werden muss. Je größer die Fenstergröße, desto ruhiger und stabiler - aber auch träger - reagiert das Netz auf Veränderungen.

`Pollrate`
: Die Pollrate in Sekunden gibt an, wie oft Olsrd den Empfangspuffer des Betriebssystems nach eingetroffenen OLSR-Nachrichten abfragt. Ein häufigeres Überprüfen erzeugt eine höhere CPU-Last. Allerdings sollte der Wert nicht zu groß eingestellt werden, da es bei einem hohen Aufkommen von OLSR-Paketen zu Pufferüberläufen im Empfangspuffer des Linuxkernels kommen kann. Der Kernel speichert nur eine bestimmte Datenmenge von eingehenden UDP-Paketen.

  Werden diese UDP-Pakete nicht rechtzeitig von Olsrd abgeholt, verwirft sie der Kernel bei einem Pufferüberlauf. Die Puffergröße ist im Linuxkernel einstellbar, so dass man mit langen Pollintervallen experimentieren kann, um die CPU-Last zu reduzieren. Es gibt eine Grundeinstellung für die Puffergröße und einen Maximalwert (`rmem_default` und `rmem_max`). Der folgende - außerhalb dieser Konfigurationsdatei - auszuführende Befehl setzt die Puffergröße auf 128 Kilobyte, dieses Kommando kann von Hand auf der Kommandozeile (in einem Terminal) mit Administratorrechten gesetzt werden:

  ```
  echo 131072 > /proc/sys/net/core/rmem_default
  ```

  Gelesen werden kann die Puffergröße mit dem Befehl:

  ```
  cat /proc/sys/net/core/rmem_default
  ```

  Hat man eine Einstellung gefunden, die bei jedem Neustart des Betriebssystems gesetzt werden soll, kann man ein Skript anlegen, welches beim Start des Betriebssystems oder beim Starten von Olsrd automatisch ausgeführt wird.

`TcRedundancy`
: Informationen, die mit Topologienachrichten versendet werden. Hier an der Redundanz zu sparen, wirkt sich negativ auf die Stabilität aus. Der Maximalwert ist 2 - alle Nachbarn mitteilen. Dieser Eintrag muss unbedingt gesetzt werden, sonst wird bei fehlendem Eintrag Null verwendet, und es gibt keine Redundanz.

`MprCoverage`
: Abdeckung mit Multipoint-Relais. Gibt an, wie viele Nachbarn ein Knoten zu Multipoint-Relais bestimmt. Hier wird ein hoher Wert gesetzt, so dass die Funktion praktisch alle Nachbarn verwendet. Durch einen Fehler im Code kommt es aber zu Abstürzen, wenn ein Wert größer 7 gesetzt wird. Auch dieser Wert muss unbedingt angegeben sein, sonst wird 1 verwendet.

`LoadPlugin`
: Olsrd-Plugins, die geladen werden sollen. Hier muss ein absoluter Pfad angegeben werden, wenn die Plugins nicht in `/lib/` oder `/usr/lib` oder in einem Verzeichnis gespeichert wurden, auf das die Variable `LD_LIBRARY_PATH` oder `/etc/ld.so.cache` verweisen. Dokumentationen zu den Plugins finden sich im Sourcecode-Verzeichnis des Olsrd unter `lib/`*`Pluginname`*. Den Plugins können/müssen mitunter Plugin-Parameter als `PlParam` mitgegeben werden. Die Plugin-Parameter stehen in einem von geschweiften Klammern eingegrenzten Konfigurationsblock zum Plugin.

`PlParam "Interval"`
: Das dynamische Gateway-Plugin `olsrd_dyn_gw.so.0.4` in der Konfigurationsdatei prüft durch Versenden von ICMP-Daten (Ping) Richtung Internet in einem bestimmten Intervall, ob der Zugang funktioniert.

  Hier wird innherhalb des Konfigurationsblocks das Prüf-Intervall angegeben. Wird das Intervall nicht definiert, geschieht die Überprüfung alle 5 Sekunden.

`PlParam "Ping"`
: Hier werden die IP-Adressen von Rechnern im Internet angegeben, die vom dynamischen Gateway-Plugin auf ihre Erreichbarkeit geprüft werden. Hier sollte man mehrere Einträge vornehmen, damit nicht wegen eines Ausfalls der Gegenstelle das Gateway seine Funktion ungewollt einstellt. Wenn mehrere IPv4-Adressen angegeben werden, werden diese in absteigender Reihenfolge per Ping getestet. Ist der erste Test erfolgreich, wird die zweite Adresse nicht geprüft.

`Interface`
: An dieser Stelle beginnt die Interface-Sektion. Hier werden die Interfaces angegeben, auf denen der OLSR-Daemon aktiv wird. Die Konfigurationsparameter des Interfaces sind in einem Konfigurationsblock zusammengefasst, der von geschweiften Klammern eingefasst wird. Mehrere Interfaces können einem Konfigurationsblock zugeordnet werden, wenn die Konfigurationsparameter gleich sind. Werden Optionen ausgelassen, verwendet OLSR die Default-Einstellungen, was in der Regel nicht sinnvoll ist.

Zunächst werden die Interfaces angegeben, die diesem Konfigurationsblock zugeordnet werden.

```
Interface "eth0" "ath0" "wlan0" "eth2"
{
...
}
```

`Ip4Broadcast`
: Die verwendete IPv4-Broadcast-Adresse. Ein sinnvolles Beispiel wäre 255.255.255.255 - diese schließt alle möglichen Netzadressen ein. Dann können sich alle Nodes auch über Subnetzgrenzen hinweg sehen. Das heißt: Verwendet ein Mesh die Netzwerkadresse 10.0.0.0/8, werden auch Nodes mit einer IP-Adresse außerhalb des Subnetzes (beispielsweise 172.29.200.0/24) über Hostrouten geroutet. Das funktioniert jedoch nur unter Linux und wenn alle Knoten im Mesh die Broadcastadresse 255.255.255.255 verwenden. Falls nicht angegeben, wird die Broadcastadresse der Interfaces verwendet. Im Normalfall auskommentiert lassen.

`Ip6AddrType`
: Verwendeter IPv6-Adressbereich (`site-local` oder `global`).

`Ip6MulticastSite`
: IPv6-Multicast-Adresse, wenn der Adressbereich `site-local` ist. Default ist `ff05::15`.

`Ip6MulticastGlobal`
: IPv6-Multicast-Adresse, wenn der Adressbereich `global` ist. Default ist `ff0e::1`.

`HelloInterval`
: Emissionsintervall für Hello-Nachrichten in Sekunden (alle Sekundenangaben sind Fließkommawerte). Werden für OLSR-Nachrichten keine Werte gesetzt, verwendet der Daemon die wenig sinnvollen RFC-Werte.

`HelloValidityTime`
: Gültigkeitsdauer der Hello-Nachrichten in Sekunden.

`TcInterval`
: Topologienachrichtenintervall in Sekunden. Falls das Mesh mehr als 100 Knoten hat, sollte das Intervall für TC-Messages vergrößert werden. Ein TCInterval-Wert von 0.5 Sekunden kann nur verwendet werden, wenn das Netz sehr klein oder der Fischaugen-Mechanismus (`LinkQualityFishEye`) aktiviert ist. Andernfalls erstickt das Netz in einer Schwemme von Topologienachrichten.

`TcValidityTime`
: Gültigkeitsdauer der Topologienachrichten in Sekunden.

`MidInterval`
: MID-Nachrichten-Intervall in Sekunden. MID-Nachrichten geben bekannt, ob ein Knoten mehrere Interfaces hat, auf denen Olsrd routet.

`MidValidityTime`
: Gültigkeitsdauer von MID-Nachrichten in Sekunden. Es schadet in keiner Weise, hier einen sehr hohen Wert anzugeben.

`HnaInterval`
: HNA-Nachrichten-Intervall in Sekunden.

`HnaValidityTime`
: Gültigkeitsdauer von HNA-Nachrichten in Sekunden. Darf ruhig sehr lang sein, damit ein Gateway nicht wegen Paketverlusts aus der Routingtabelle fällt.

`LinkQualityMult`
: Soll eine bestimmte Route manuell bevorzugt oder ignoriert werden, können hier ETX-Link-Quality-Werte über einen Multiplikator beeinflusst werden. In diesem Beispiel würde die Route über 192.168.0.1 schlechter erscheinen als sie ist. Durch das Multiplizieren mit dem Faktor 0.5 wird der LinkQuality-Wert klein (schlecht) und der ETX-Wert entsprechend hoch.

  Auch der umgekehrte Weg ist möglich. Hier werden alle Linkstrecken um den Faktor 0.7 schlechter gerechnet. Im Folgenden werden zwei bestimmte Routen besser gestellt:

```
LinkQualityMult default 0.7
LinkQualityMult 10.143.143.143 1.0
LinkQualityMult 10.143.143.144 1.0
```

An dieser Stelle ist nun eine gültige Konfigurationsdatei abgeschlossen. Falls weitere Interfaces mit abweichenden Parametern konfiguriert werden sollen, können weitere Interface-Blöcke hinzugefügt werden. In der Konfigurationsdatei folgt noch die Konfiguration eines Tunnel-Interface. Tunnel sind beliebt zur Kopplung von isolierten kleineren OLSR-Netzen, aber auch als Backbone innerhalb eines stadtweiten Netzes. Auf dem Tunnel-Interface kann Olsrd sehr langsam und bandbreitenschonend konfiguriert werden.

3.3.4 Erster Start des Daemon

Ist die Konfigurationsdatei fertig editiert, genügt auf der Kommandozeile die Ausführung des Befehls `olsrd` mit Administratorrechten. Optional -

zumal beim ersten Start nach umfangreichen Arbeiten an der Konfiguration – kann der Daemon unter Angabe eines Debuglevels gestartet werden:

```
olsrd -d 1
```

Der OLSR-Daemon ist bereits bei Debuglevel 1 sehr gesprächig und liefert eine Flut von Informationen, wenn die Routensuche funktioniert. Höhere Debuglevel sind nur für Softwareentwickler interessant, die tatsächlich Informationen über die internen Abläufe des Daemon gewinnen wollen, um Programmierfehler zu finden. Die Ausgaben des Olsrd rauschen – außer bei sehr kleinen Meshnetzen – schon bei Debuglevel 1 im Terminalfenster durch und erzeugen eine hohe Systemlast.

Funktioniert das Routing nicht oder ist ein Fehler in der Konfigurationsdatei aufgetreten (Syntaxfehler in der Konfigurationsdatei, nicht vorhandenes oder nicht konfiguriertes Netzwerkinterface), erzeugt der Daemon dagegen bei einem schweren Fehler wenige Ausgaben, die man problemlos beobachten kann. Schlägt dagegen lediglich das Laden eines Plugins fehl, oder funktioniert von mehreren Interfaces eines nicht, läuft die Ausgabe viel zu schnell durch den Bildschirm, um noch lesbar zu sein. In diesem Fall kann man die Ausgabe des Daemon in eine Datei schreiben oder sie über eine Pipe einem Pager wie `less` übergeben:

```
olsrd -d 1 2>&1 |cat > /tmp/olsrd.log
```

oder eben

```
olsrd -d 1 2>&1 |less
```

Außer zur Fehlersuche sind die Debuglevel des Olsrd deshalb kaum geeignet, um Auskunft über den Zustand des Meshroutings zu erhalten. Möchte man Informationen über den Zustand des Mesh beobachten, ist es empfehlenswert, das Httpinfo-Plugin des Olsrd anzuwenden, das im Abschnitt 3.4.2 beschrieben wird.

Wurde der OLSR-Daemon mit Debuglevel 1 in einem kleinen Mesh gestartet, sollte die Ausgabe, wenn alles funktioniert, wie folgt aussehen (Topologieliste gekürzt):

```
--- 21:57:54.15 ------------------------------------------ LINKS

IP address       hyst   LQ     lost   total  NLQ    ETX
104.130.30.1     0.000  0.600  40     100    0.310  5.38
104.130.30.29    0.000  0.920  8      100    0.910  1.19

--- 21:57:54.15 ---------------------------------------- NEIGHBORS
```

```
IP address       LQ     NLQ    SYM   MPR   MPRS  will
104.130.30.29    0.920  0.910  YES   YES   NO    7
104.130.30.1     0.600  0.310  YES   YES   NO    3

--- 21:57:54.02156644 ---------------------- TWO-HOP NEIGHBORS

IP addr (2-hop)  IP addr (1-hop)  TLQ
104.130.1.243    104.130.30.1     0.070
104.130.1.67     104.130.30.1     0.021
104.192.192.199  104.130.30.1     0.000
104.130.30.29    104.130.30.1     0.149
104.130.30.1     104.130.30.29    0.675

--- 21:57:54.15 ---------------------------------------- TOPOLOGY

Source IP addr   Dest IP addr     LQ     ILQ    ETX
104.129.105.1    104.129.105.2    0.886  0.949  1.19
104.129.105.1    104.129.108.3    0.220  0.047  96.76
104.129.105.1    104.129.105.10   0.333  0.929  3.23
104.129.105.1    104.129.0.190    0.918  0.929  1.17
```

Die Ausgabe unter der Rubrik `LINKS` listet alle direkten Nachbarn auf. Der Wert für `hyst` ist immer Null, da die Hysterese ausgeschaltet ist. `LQ` (Link-Quality) gibt das Verhältnis von verlorenen zu empfangenen Hello-Paketen an, die von dem angegebenen Nachbarn gesendet werden. Der Nachbarknoten mit der IP-Adresse 104.30.30.22 hat beispielsweise 60 von 100 Hello-Nachrichten erfolgreich an unseren OLSR-Router verschickt. Der Wert `NLQ` (Neighbor Link Quality) gibt dessen Statistik über die von unserem Knoten verschickten Hello-Nachrichten an.

Die Rubrik `Neighbors` enthält zusätzliche Informationen über die direkten Nachbarn, die allerdings weitgehend redundant sind. Die Angabe `SYM` zeigt an, ob die Verbindung zum Nachbarn symmetrisch ist, d. h. in beide Richtungen funktioniert. Diese Information ist auch anhand der LQ/NLQ-Werte ersichtlich. Solange weder der LQ- noch der NLQ-Wert Null ist, gilt ein Nachbar als symmetrisch. Die Spalte `MPR` gibt an, ob es sich bei dem Nachbarn um ein Multipoint-Relais handelt. `MPRS` gibt an, ob sich der Nachbarknoten unseren Router als Multipoint-Relais ausgesucht hat. `Will` zeigt die Willingness (Bereitschaft) eines Knoten an, fremde Datenpakete zu routen.

`TOPOLOGY` listet die LQ/NLQ-Werte von allen direkten Funklinks zwischen allen Knoten im gesamten Mesh auf. Diese Liste wird deshalb schnell sehr lang, wenn das Mesh mehr als nur ein paar Knoten hat.

Der Olsrd trägt nun für sämtliche Meshknoten im Netzwerk, zu denen sich eine Route berechnen lässt, eine Hostroute in die Routingtabelle des Betriebssystems ein. Dabei erscheint jeweils einer der direkten Nachbarn als Gateway zu dem entfernten Knoten. Unter Linux kann die Routingtabelle beispielsweise mit dem Befehl `route -n` ausgegeben und überprüft werden, der folgende, hier auszugsweise wiedergegebene Ausgabe erzeugt:

```
IBM-R50:/tmp# route -n
Kernel IP routing table
Destination     Gateway         Genmask          Flags Metric Ref Use Iface

104.19.19.18    104.130.30.29   255.255.255.255 UGH   15     0      0 ath0
104.129.0.35    104.130.30.29   255.255.255.255 UGH   8      0      0 ath0
104.19.19.19    104.130.30.29   255.255.255.255 UGH   14     0      0 ath0
104.131.83.2    104.130.30.29   255.255.255.255 UGH   5      0      0 ath0
104.129.1.1     104.130.30.29   255.255.255.255 UGH   6      0      0 ath0
104.129.0.1     104.130.30.29   255.255.255.255 UGH   6      0      0 ath0
104.130.30.1    104.130.30.29   255.255.255.255 UGH   2      0      0 ath0
104.130.30.29   0.0.0.0         255.255.255.255 UH    1      0      0 ath0
192.168.212.0   0.0.0.0         255.255.255.0   U     0      0      0 eth1
104.0.0.0       0.0.0.0         255.0.0.0       U     0      0      0 ath0
0.0.0.0         104.130.30.29   0.0.0.0         UG    5      0      0 ath0
```

In der Spalte `Destination` sind die erreichbaren Ziele aufgelistet. Hostrouten sind erkennbar an der `Genmask` (Netzmaske) 255.255.255.255 und am Flag H für Hostroute. Die Route mit der Destination 0.0.0.0 und der Netzmaske 0.0.0.0 in der untersten Zeile ist eine Defaultroute. Sie wird verwendet für alle Ziele, die nicht in einer weiter oben liegenden Zeile der Routingtabelle gelistet sind. Eine Defaultroute kann – muss aber nicht – ins Internet führen. Von OLSR-Routern, die eine Route mit dem Ziel 0.0.0.0 ankündigen, wird jedoch erwartet, dass sie Internetzugang bereitstellen. Der OLSR-Daemon setzt für jede Route eine Metrik, die der Anzahl von Hops bis zum Ziel entspricht. Im Fall der aufgelisteten Defaultroute bedeutet das, dass das Internet-Gateway im Mesh fünf Hops entfernt liegt.

Ob beispielsweise der Internetzugang funktioniert, kann geprüft werden, indem eine bekannte IP-Adresse eines Computers im Internet mit dem Befehl `ping` getestet wird.

```
IBM-R50:/tmp# ping 141.1.1.1
PING 141.1.1.1 (141.1.1.1) 56(84) bytes of data.
64 bytes from 141.1.1.1: icmp_seq=1 ttl=51 time=1167 ms
64 bytes from 141.1.1.1: icmp_seq=2 ttl=51 time=288 ms
64 bytes from 141.1.1.1: icmp_seq=3 ttl=51 time=93.5 ms
```

Natürlich kann statt der IP-Nummer auch der Domainname einer bekannten Domain im WWW angegeben werden. Wenn eine IP-Adresse mit Ping erreicht werden kann, aber ein Ping zu einem Domainnamen nicht gelingt, ist entweder kein Domain-Name-Server im System konfiguriert oder dieser ist aus dem Mesh nicht erreichbar. In einem OLSR-Mesh gibt es keine Autokonfiguration von Meshknoten mit IP-Adresse, Netzmaske, Domain-Name-Server-Adresse über das DHCP-Protokoll. Diese Konfigurationsparameter müssen manuell im Computer eingetragen werden.

Um die Eintragung des Gateways in das System muss man sich dagegen nicht kümmern, diese Aufgabe wird vom OLSR-Daemon übernommen. Im

Gegenteil: Ein Default-Gateway sollte nicht eingetragen sein, wenn man über das Mesh ins Internet geroutet werden möchte. Findet das Betriebssystem mehrere Gatewayeinträge zur Destination 0.0.0.0 vor, wird das Gateway mit der kleinsten Metrik verwendet. Ein manuell eingetragenes Default-Gateway hat normalerweise eine Metrik von Null, ein Default-Gateway von OLSR hat eine Metrik von mindestens Eins, da OLSR mindestens einen Hop zum Gateway benötigt. In diesem Fall wird deshalb immer das manuell eingetragene Gateway verwendet. Ist dieses nicht erreichbar, funktioniert der Internetzugang über das Mesh nicht.

Der Verlauf einer Route lässt sich mit dem Befehl

```
traceroute -n   Zieladresse
```

unter Linux oder

```
tracert Zieladresse
```

unter Windows überprüfen. Die Option `-n` bewirkt, dass `traceroute` die IP-Adressen der Zwischenstation ausgibt und nicht versucht, deren Domainnamen herauszufinden, was bedeutend schneller geht.

fftrace

Zur Analyse von Olsr-Routen gibt es das Programm fftrace (Freifunk-Traceroute), das für das Überwachen und Verfolgen von OLSR-Routen entwickelt wurde. fftrace zeigt ETX-Werte an und kommt besser als `mtr` oder `traceroute` mit dem häufigen Umschalten von Routen zurecht[3].

3.4 Wichtige Plugins für Olsrd

Die Funktionalität von Olsrd lässt sich durch Plugins erweitern. Das dynamische Gateway-Plugin wurde im vorherigen Abschnitt bereits erwähnt. Mehrere `LoadPlugin`-Blöcke lassen sich in der Konfigurationsdatei `/etc/olsrd.conf` nacheinander einfügen. Sind die Plugins nicht in einem Verzeichnis installiert, in dem der Olsrd beim Start nach installierten Plugins sucht, muss ein absoluter Dateipfad angegeben werden.

Unter Windows sucht Olsrd in dem Verzeichnis, aus dem der Daemon gestartet wurde, nach den Plugins. Alternativ können diese Dateien (sie haben die Endung `.dll`) in den Systemordner `c:\Windows\system32` kopiert werden.

Kann ein Plugin nicht geladen werden, gibt der Daemon eine Warnmeldung aus, wenn er mit einem Debuglevel größer oder gleich 1 aufgerufen wurde.

[3] `http://downloads.open-mesh.net/fftrace`

3.4.1 DotDraw

Dieses Plugin gibt über TCP-Port 2004 Topologieinformationen aus. Mit diesen Daten lassen sich zwei- und dreidimensionale Topologiegrafiken erzeugen. Die ausgegebenen Daten sind zu den Visualisierungsprogrammen Graphviz und olsrs3d kompatibel.

Zum Einrichten des Plugins muss die Datei `/etc/olsrd.conf` geringfügig erweitert werden. Das Plugin kann ohne zusätzliche Parameter gestartet werden; es genügt, einen leeren Block mit zwei geschweiften Klammern anzuhängen. In den Grundeinstellungen kann nur lokal auf die Ausgabe des Plugins zugegriffen werden. Wer die Ausgaben von einem entfernten Computer abrufen will, muss den Zugriff von dessen IP-Adresse explizit erlauben. Unter Linux:

```
LoadPlugin "olsrd_dot_draw.so.0.3"
        {
                PlParam     "accept" "192.168.100.1"
                }
```

und unter Windows:

```
LoadPlugin "olsrd_dot_draw.dll"
        {
        }
```

Ob das Plugin ordnungsgemäß arbeitet, kann mit einem Connect per Telnet oder über einen Webbrowser getestet werden.

```
telnet IP-Adresse-des-Olsr-Routers 2004
```

Telnet sollte nun lesbare Informationen in ASCII-Text ausgeben. In einem Webbrowser kann einfach die URL der Maschine mit dem DotDraw-Plugin zusammen mit der Portnummer geöffnet werden, also beispielsweise `http://localhost:2004`

Meist arbeitet das DotDraw-Plugin auf einem kleinen Meshrouter, von dem ein leistungsfähigerer Rechner über Telnet die Daten abholt und daraus die Topologiegrafiken generiert. Viele Freifunk-Communities bieten kontinuierlich aktualisierte Topologiegrafiken auf ihren Webseiten an.

Toplogievisualisierung

Es gibt ein Perlscript[4], das automatisch die Informationen von DotDraw abholt und daraus zweidimensionale Topologiegraphen erzeugt. Dazu müssen die Programme Graphviz, Perl und ImageMagick (optional) installiert

[4] `http://meshcube.org/nylon/utils/olsr-topology-view.pl`

sein. Das Skript erzeugt Grafikdateien in unterschiedlichen Formaten wie JPEG, PNG, Scalable Vector Graphics, PostScript usw.

```
perl olsr-topology-view.pl --server IP-Adresse des Olsr-Routers --noshow
```

Abbildung 3.2: Zweidimensionale Topologievisualisierung mit DotDraw[6]

Optisch ansprechender ist die rotierende dreidimensionale Visualisierung über das Tool olsrs3d[7] von Simon Wunderlich und Marek Lindner. Olsrs3d ist eine Applikation für den dreidimensionalen Desktop s3d auf der Basis von OpenGL. Zur Visualisierung eines Mesh mit mehr als ein paar Dutzend Knotenpunkten wird ein schneller Rechner mit einer leistungsfähigen Grafikkarte benötigt.

3.4.2 Httpinfo

Wird dieses Plugin geladen, startet Olsrd einen eigenen, minimalistischen Webserver und zeigt detaillierte Informationen über den Status des Daemon auf mehreren HTML-Seiten an. Auf diese Weise kann die Funktion des Routers mit einem Browser lokal und über das Netzwerk beobachtet werden.

Die Formatierung der generierten HTML-Seiten ist leider (noch) nicht perfekt, die Beschriftung der einzelnen Spalten ist verschoben.

Aus Sicherheitsgründen kann der Zugriff nur von bestimmten Rechnern oder Netzwerken auf den Webserver erlaubt werden, da dieser potentiell ein Einfallstor für Angriffe gegen den Router darstellt. In der Beispielkonfiguration ist nur der lokale Zugriff auf den Webserver erlaubt. Weitere Konfigurationsbeispiele für den Zugriff von Host 80.23.53.22, Netzwerk 10.0.0.0/8 und Netzwerk 192.168.0.0/24 sind auskommentiert.

[6] Ausschnitt der Grafik `http://www.the-mesh.org/dotdraw/olsr_dot_draw.jpg`.
[7] `http://s3d.berlios.de/`

Abbildung 3.3: Httpinfo

Einrichtung unter Linux:

```
LoadPlugin "olsrd_httpinfo.so.0.1"
{
        # Die Portnummer, auf der der Webserver arbeitet
            PlParam     "port"    "8080"
        # In der Grundeinstellung ist nur 127.0.0.1
             # (lokale Maschine auf der Olsrd läuft)
        # der Zugriff auf den Webserver erlaubt.
        PlParam      "Host"    "127.0.0.1"

        # Weitere Knoten können hinzugefügt werden.
        # PlParam     "Host"     "10.0.0.5"

        # Der Zugriff kann auch für ganze Adressbereiche
        # gestattet werden. Mehrere Einträge sind erlaubt.

        # PlParam     "Net"      "192.168.1.0 255.255.255.0"
}
```

Unter Windows können prinzipiell dieselben Plugin-Parameter verwendet werden wie unter Linux. Es ist auch hier ein Sicherheitsrisiko, den Webserver für den Zugriff von fremden Maschinen zuzulassen.

```
LoadPlugin "olsrd_httpinfo.dll"
{
        PlParam     "port"    "8080"
        PlParam             "Host"    "127.0.0.1"
}
```

Wird die Beispielkonfiguration auf einem Arbeitsplatzrechner eingesetzt, können die Statusseiten des Olsrd unter der URL `http://127.0.0.1:8080` aufgerufen werden.

Die von Httpinfo angezeigten Informationen sind sehr hilfreich im praktischen Betrieb eines Mesh. So lässt sich überprüfen, ob die Verbindung zu den direkten Nachbarn brauchbar ist, oder die gesamte Topologie betrachten, da OLSR diese Informationen in der lokalen Datenbank speichert. Sollte eine normalerweise stabil funktionierende Route plötzlich nicht mehr funktionieren, kann anhand der Topologieinformationen nachgeprüft werden, ob alle Funkstrecken auf der Route richtig funktionieren. Außerdem werden sämliche HNA-Announcements dargestellt.

3.4.3 Nameservice

Dieses Plugin verteilt DNS-Informationen über OLSR. OLSR-Knoten können damit ihren Namen und die Namen anderer Rechner mitteilen, zu denen Routen über eigene HNA-Nachrichten annonciert werden. Also beispielsweise DNS-Einträge für Rechner am LAN-Port eines OLSR-Routers, die statische Adressen haben und aus dem Mesh über eigene HNA-Ankündigungen erreichbar sind. Auch ein „echter" DNS-Server, der Internetadressen auflöst, kann anderen OLSR-Knoten mitgeteilt werden. Allerdings hat das Plugin noch ein paar Haken und Ösen – ist also noch nicht so richtig komfortabel und ausgegoren.

In jedem Linux/Unix-System gibt es die Datei `/etc/hosts`, diese enthält eine Tabelle, in der Hostnamen statisch zu IP-Adressen zugeordnet werden können. Das Nameservice-Plugin erzeugt eine `hosts`-Datei und fügt DNS-Einträge aus dem OLSR-Mesh hinzu. Damit die vom Nameservice erzeugte Datei vom Betriebssystem genutzt wird, muss sie unter Linux oder BSD unter `/etc/hosts` stehen oder mit einem symbolischen Link an diese Stelle verlinkt werden. Im einfachsten Fall lässt man das Plugin die Datei neu anlegen und konfiguriert es entsprechend.

Der Nachteil ist, dass bei jedem Start des Olsrd eine bereits vorhandene Datei `/etc/hosts` von dem Plugin überschrieben wird. Wurden dort manuell eigene Einträge angelegt, gehen diese beim Einsatz des Plugins verloren. Die Default-Einstellung des Plugins ist daher das Anlegen einer Datei (Linux: `/var/run/hosts_olsr`, Windows: `C:\WINDOWS\hosts_olsr`). Allerdings müssen die Anwender dann selbst Wege finden, diese Einträge dem jeweiligen System verfügbar zu machen. Linux sucht nur in der Datei `/etc/hosts` nach statisch zugeordneten Hostnamen. Sind keine wichtigen Einträge in `/etc/hosts`, darf das Plugin natürlich diese Datei einfach überschreiben.

Ebenso werden durch das Nameservice-Plugin die zu verwendenden DNS-Server unter Linux/Unix in die Datei `/etc/resolv.conf` eingetragen. Hier

werden die drei nächstgelegenen DNS-Server hinzugefügt, die im Mesh am besten erreichbar sind.

Soll der Inhalt von `/etc/hosts` und `/etc/resolv.conf` erhalten bleiben, wäre beispielsweise ein Shellskript eine Lösungsmöglichkeit. Dieses Skript könnte beim Start des OLSR-Daemon eine Sicherheitskopie der vorhandenen Datei `/etc/hosts` anlegen und diese nach dem Beenden von Olsrd wieder herstellen.

Auf die gesammelten DNS-Informationen kann ein lokaler DNS-Server wie beispielsweise der minimalistische `dnsmasq` zugreifen und sie DNS-Clients, die keine Mesh-Clients sind, direkt zur Verfügung stellen. Wenn das Plugin seine gewonnenen DNS-Einträge in `/var/run/hosts_olsr` schreibt, kann in `dnsmasq.conf`, der Konfigurationsdatei von `dnsmaq`, diese Zeile hinzugefügt werden: `addn-hosts=/var/run/hosts_olsr`. So kann `dnsmasq` die DNS-Einträge aus dem Nameservice-Plugin verwenden, muss dazu aber neu gestartet werden. Einrichtung des Nameservice-Plugins unter Linux:

```
LoadPlugin "olsrd_nameservice.so.0.2"
{
        # Der Name dieses OLSR-Knotens, der dem Mesh mitgeteilt wird:
        PlParam "name" "Name-dieser-Maschine"

        # Lokale Knoten, die über HNA announciert werden:
        PlParam "10.134.133.56" "Sonne"
        PlParam "10.134.133.57" "Mond"

        # DNS-Einträge werden zu dieser Datei hinzugefügt.
        PlParam "hosts-file" "/etc/hosts"

        # Zusätzliche Datei für DNS-Einträge statt /etc/hosts
        # verwenden. Dadurch wird verhindert, dass die originale
        # Hosts-Datei überschrieben wird und bereits selbst angelegte
        # Einträge verlorengehen.
        #PlParam "add-hosts" "Datei mit Pfadangabe"

        # In diese Datei werden die drei nächstgelegenen DNS-Server
        # eingetragen, welche den besten ETX-Wert haben.
        PlParam "resolv-file" "/etc/resolv.conf"

        # Jedem DNS-Eintrag von OLSR kann ein Suffix angehängt werden.
        # In der Grundeinstellung wird das nicht gemacht.
        # PlParam "suffix" ".olsr"

        # Über diesen Eintrag wird ein 'echter' DNS-Server annonciert.
        # Wird der Eintrag leer gelassen ("") wird die IP-Adresse dieses
        # OLSR-Knotens verwendet.
        # PlParam "dns-server" "IP-Adresse des DNS-Servers"

        # Das Intervall in dem DNS-Informationen an andere OLSR-Knoten
        # verschickt werden (in Sekunden). In der Grundeinstellung werden
```

```
        # 120 Sekunden verwendet.
        # PlParam "interval" "Anzahl Sekunden"

        # Wie lange sind über das Plugin empfangene DNS-Informationen
        # gültig (in Sekunden)? Grundeinstellung 3600 Sekunden (1 Stunde)
        # PlParam "timeout" "3600"
}
```

Unter Windows kann das Plugin dem Betriebssystem keine empfangenen DNS-Servereinträge hinzufügen. Abgesehen davon funktioniert es wie unter Linux. Hosteinträge landen in den Grundeinstellungen in `hosts_olsr` im Windows-Systemordner `C:\WINDOWS` oder `C:\WINNT`. Möchte man die gefundenen Hosteinträge direkt verwenden, kann das Plugin sie in die Datei `C:\Windows-System-Ordner\system32\drivers\etc\hosts` schreiben. Manche Ad-Blocker-Programme verwenden diese Datei jedoch, um Zugriffe auf unerwünschte Webseiten zu blockieren, indem sie deren Adressen über die Hosts-Datei umleiten. Der Einsatz des Plugins wird die Arbeitsweise dieser Programme behindern.

```
LoadPlugin "olsrd_nameservice.dll"
{
        PlParam "name" "Name-dieser-Maschine"
        PlParam "hosts-file" "c:\WINNT\system32\drivers\etc\hosts"
}
```

Die weiteren Plugin-Parameter entsprechen dem Einrichtungsbeispiel für Linux.

3.5 Typische Probleme mit OLSR

3.5.1 Gateway-Switching

Olsrd schaltet unter Umständen häufig das verwendete Internet-Gateway um, wenn es in einem OLSR-Mesh mehrere Internet-Gateways gibt. Ein Gateway-Client kann unter OLSR nicht entscheiden, über welches Internet-Gateway er letzten Endes ins Internet geroutet wird, solange das Gateway kein direkter Nachbar ist. Jeder OLSR-Router auf einem Multi-Hop-Pfad zum Internet entscheidet unabhängig von den Wünschen des Gateway-Clients, zu welchem Gateway er gerade die Internetpakete schickt. Dabei wird jeweils das günstigste Gateway gewählt, zu dem gerade die beste Route besteht. Die Qualität von Routen verändert sich in einem Mesh jedoch beständig. Deshalb geschieht es in einem Mesh mit mehreren Gateways relativ häufig, dass das Internet-Gateway von OLSR umgeschaltet wird.

Die meisten Internet-Gateways verwenden NAT (Network Address Translation), d. h. sie maskieren die private IP-Adresse des Gateway-Clients und

kommunizieren mit dem Server im Internet über ihre öffentliche IP-Adresse. Der Server empfängt alle Anfragen des Gateway-Clients mit der öffentlichen IP-Adresse des Gateways und sendet seine Antworten an diese Adresse zurück.

Immer wenn OLSR das Internet-Gateway wechselt, brechen die bestehenden Verbindungen zusammen, da der Server im Internet nicht weiß, dass der Client nun über eine andere öffentliche Adresse mit ihm zu kommunizieren versucht. Die bestehende Session ist aus Sicht des Servers unterbrochen, und der Client muss eine neue Session aufbauen. Häufiges Umschalten des Gateways macht Anwendungen, die ein verbindungsorientiertes Protokoll einsetzen, nahezu unbenutzbar: Downloads brechen immer wieder ab, VoIP-Gespräche enden abrupt, SSH-Verbindungen setzen plötzlich aus.

Sitzt man mit OLSR zwischen zwei oder mehreren annähernd gleich gut erreichbaren Gateways, kann das Umschalten mehrmals pro Minute vorkommen. Da jeder OLSR-Router individuell sich für die Weiterleitung zu einem anderen Gateway entscheiden kann, hat man bei einer Internetverbindung über mehrere Zwischenhops keine Möglichkeit, das zu beeinflussen. Eine Lösung wäre, dass man den Gatewaybetreiber persönlich kennenlernt und mit ihm zusammen manuell einen IP-Tunnel (z. B. mit OpenVPN) aufbaut. Aber was soll man tun, wenn dieses Gateway gerade nicht funktioniert oder nicht erreichbar ist?

Einzelne Freifunk-Communities haben das Problem gelöst, indem sie den Internettraffic von jedem Gateway im Mesh zu einem gemeinsamen Proxy/Gateway im Internet weiterreichen, der in das Routing einbezogen wird. Alle Internetanfragen werden über den Proxy geroutet und nicht vorher durch Network Address Translation maskiert. Der Proxy weiß daher immer, über welches Gateway er zuletzt von einem OLSR-Knoten angesprochen wurdel, und versendet die Anfragen jeweils an das Gateway, hinter dem der OLSR-Knoten auf seine Antwort aus dem Internet wartet. Diese Lösung ist einigermaßen elegant, setzt aber einen erhöhten Konfigurationsaufwand voraus.

Nachteilig ist außerdem, dass der gesamte Datenverkehr eines stadtweiten Netzwerks über einen einzelnen Server abgewickelt wird – dieser muss den gesamten anfallenden Datenverkehr entgegennehmen und weiterversenden. Dadurch fallen zusätzliche Kosten an, wenn der Traffic nicht kostenlos ist oder pauschal abgerechnet wird. Zudem bricht der gesamte Internetverkehr aus dem Mesh zusammen, wenn der Proxy/Gateway ausfällt. Außerdem kann der gesamte Datenverkehr vom/zum Mesh ins Internet an einer einzigen Stelle überwacht und abgehört werden.

Dieses Problem haben die Entwickler bei B.A.T.M.A.N. durch das automatische Aushandeln von Tunnels von vornherein ausgeräumt.

3.5.2 Typische Probleme beim ersten Einsatz

Voraussetzung für den Kontakt zum Meshnetzwerk ist eine gültige IP-Adresse für jedes Interface, auf dem der Mesh-Routingdaemon arbeitet. Dazu gehören auch die zur Adresse des Netzwerks passende Netzmaske, die Broadcastadresse und ein gültiger DNS-Server. Ein beliebter Fehler ist eine ungültige Broadcastadresse. Stimmt die Broadcastadresse nicht, ignorieren die Meshknoten gegenseitig ihre OLSR-Pakete – diese werden ausnahmslos per Broadcast verschickt. In einem Mesh gibt es prinzipiell keinen DHCP-Server (Ausnahme: für die lokalen Clients eines Meshrouters, wenn diese nicht am Routing beteiligt sind). Das Eingabefeld **Gateway** muss beim Einsatz der Protokolls unbedingt leer gelassen werden, es darf kein Default-Gateway angegeben werden. Diese Einstellung nimmt der Routing-Daemon dynamisch vor. Auch die WLAN-Karte(n) müssen für die Verbindung zur Ad-Hoc-Zelle richtig konfiguriert sein.

Bei Problemen sollte man zunächst prüfen, ob eine Firewall aktiv ist. Ist das der Fall, sollte sie testweise deaktiviert werden, um auszuschließen, dass die OLSR-Pakete von der Firewall blockiert werden. OLSR verwendet UDP-Port 698, eingehende und ausgehende UDP-Verbindungen müssen deshalb durch die Firewall erlaubt werden.

Einfaches Internet-Gateway

Möchte man für einen Test einfach und schnell ein OLSR-Gateway mit einem PC oder Notebook unter Linux einrichten, muss auf diesem Computer im Normalfall NAT (Network Address Translation) eingerichtet werden. Ansonsten ist auch nach dem Aktivieren des Dynamischen Gateway-Plugins keine Verbindung aus dem OLSR-Netz ins Internet möglich. Ist keine Firewall eingerichtet, lässt sich NAT durch das Ausführen dieser beiden Befehle aktivieren. Eventuell bereits vorhandene Firewallregeln werden dazu deaktiviert. `eth1` und `192.168.1.1` sind in diesem Beispiel das Netzwerkinterface und dessen IP-Adresse, über das das Gateway mit dem Internet verbunden ist. Diese beiden Parameter müssen der jeweiligen Situation angepasst werden.

```
iptables -F; iptables -t nat -F; iptables -t mangle -F
iptables -t nat -A POSTROUTING -o eth1 -j SNAT --to 192.168.1.1
```

3.5.3 Probleme mit WiFi

Allzu häufig machen WLAN-Karten beziehungsweise deren Treiber und/oder Firmware Ärger. Ein alltägliches Problem ist bei Meshnetzen das sogenannte Cell-Splitting oder IBSS-ID-Splitting. Ein eindeutiges Indiz für Cell-Splitting ist folgendes: Obwohl alle Parameter bei der IP-Konfiguration

stimmen und auch die Funk-Konfiguration der WLAN-Karten korrekt ist, sehen sich die WLAN-Karten gegenseitig nicht.

Solange die Computer sich in direkter, unmittelbarer Funk-Reichweite befinden – *link-local* sind –, muss es immer möglich sein, dass alle Rechner auch ohne Routingprotokoll miteinander kommunizieren. Bei Problemen sollte zuerst geprüft werden, ob die einzelnen Computer sich gegenseitig anpingen können. Klappt das nicht, sollte zunächst eine Firewall verdächtigt werden. Auf Windows-PCs blockieren Firewalls nicht selten eingehende ICMP-Pakete (Ping-Pakete) aus „Sicherheitsgründen".

Wenn eine Firewall als Fehlerursache ausgeschlossen ist, liegt das Problem wahrscheinlich in einer fehlerhaften Funktion von WLAN-Karten im Ad-Hoc-Modus, die sehr häufig auftreten. Unter Linux kann die Cell-ID durch den Befehl

```
iwconfig Interface
```

überprüft werden. Tauchen bei der Angabe der Cell-ID auf den Computern unterschiedliche Angaben auf, ist eindeutig Cell-Splitting die Ursache, wenn alle anderen Angaben (Kanal, Modus, kein Schreibfehler in der ESSID) stimmen. In diesem Fall liegt das Problem in der WLAN-Karten-Firmware und/oder den Treibern. Wie man diesen Problemen aus dem Weg gehen könnte zeigt Kapitel 5.

B.A.T.M.A.N. – Better Approach To Mobile Ad-Hoc Networking

Ausgehend von den Erfahrungen mit Freifunk-OLSR begannen die Entwickler aus der Freifunk-Community im März 2006 in Berlin damit, ein neues Routingprotokoll für drahtlose Meshnetzwerke zu entwickeln.

Alle bisher bekannten Routingalgorithmen versuchen, Routen entweder zu berechnen (proaktive Verfahren) oder sie dann zu suchen, wenn sie gebraucht werden (reaktive Verfahren). Das neue Protokoll B.A.T.M.A.N. berechnet oder sucht im Gegensatz zu diesen Protokollen keine Routen – es erfasst lediglich, ob Routen zu anderen Knoten existieren und überwacht ihre Qualität. Dabei interessiert es sich nicht dafür, wie eine Route verläuft, sondern ermittelt lediglich, über welchen direkten Nachbarn ein bestimmter Netzwerkknoten am besten zu erreichen ist, und trägt diese Information proaktiv in die Routingtabelle ein.

In der IP-Routingtabelle im Betriebssystem eines Computers ist grundsätzlich nur der nächstgelegene Router (oder ein Gateway) zu einem Ziel ein-

getragen. Ein Netzwerkknoten muss nur den jeweils nächsten direkt erreichbaren Zugangspunkt kennen, der mit dem Weiterleiten von IP-Paketen zu einem Ziel beauftragt werden kann. Kenntnisse über Zwischenstationen sind für den Absender irrelevant. Dieser hat üblicherweise auch keine Möglichkeit, den Weg seiner IP-Pakete zu beeinflussen.

Wichtig ist lediglich, dass auch der jeweils nächste Router wieder das richtige Gateway zum Ziel kennt. Solange diese Kette nicht unterbrochen wird (oder das IP-Paket im Kreis herumschwirrt, bis die Time-To-Live abgelaufen ist) erreichen die Daten ihr Ziel. Bei einem verbindungslosen Protokoll wie UDP, welches Daten nur in eine Richtung schickt, ohne eine Bestätigung (Acknowledgement) von der Empfängerseite zu erwarten, reicht es aus, wenn eine Route nur in einer Richtung funktionsfähig ist. Protokolle wie TCP dagegen erwarten eine Antwort von der Gegenseite – deshalb muss das Weiterreichen der IP-Pakete auch in der Gegenrichtung funktionieren. Es muss also nicht nur eine Route in eine Richtung, sondern auch eine Route in der Gegenrichtung existieren – was Anfänger oft vergessen, wenn sie von Hand Routingtabellen editieren. Üblicherweise nehmen IP-Pakete für den Hin- und Rückweg die gleiche Route, das ist aber nicht unbedingt notwendig.

OLSR – oder Link-State-Routing im Allgemeinen – wendet ein aufwendiges Verfahren an, um diese Vorgaben umzusetzen. Die einzige Entscheidung, die ein Link-State-Router beim Versenden oder Weiterleiten von Daten treffen kann, betrifft ebenfalls lediglich die Auswahl des nächsten Routers, der über eine direkte Verbindung per Funk oder per Kabel mit dem Weiterreichen des IP-Pakets beauftragt werden kann. Aber um diese einfache Entscheidung treffen zu können, treiben Link-State-Verfahren großen Aufwand: Das gesamte Netz wird mit Topologieinformationen geflutet, alle Routen von jedem Endpunkt des Graphen zu jedem anderen Endpunkt werden berechnet. Am Ende führt gerade die Tatsache, dass sich jeder einzelne Knoten ein eigenes Gesamtbild des Netzwerks errechnet, zu Unstimmigkeiten und in der Folge zu Kreisrouten. Die Topologiedatenbanken in einem drahtlosen Mesh mit einer nennenswerten Anzahl von Routern sind praktisch nie synchron.

Link-State-Protokolle in drahtgebundenen Netzen wie das verbreitete Protokoll OSPF vermeiden Kreisrouten, indem sie sicherstellen, dass die Topologiedatenbanken in allen Routern wirklich identisch sind. Die dazu verwendeten Verfahren versagen in einem drahtlosen Netzwerk. Bevor die Datenbanken in allen Routern tatsächlich synchronisiert sind, sind sie hoffnungslos veraltet und damit nutzlos.

Die Kenntnis über die Zwischenstationen einer Route ist nur dann relevant, wenn der Absender die Route vorgibt. In IP-basierten Netzen ist das ein Sonderfall; dieses Verfahren wird als Sourcerouting bezeichnet – die Informationsquelle (Source) gibt die Route vor. Da die Übersicht über die Netzwerktopologie in einem Mesh mit zunehmender Entfernung immer

unschärfer wird, ist eine solche Vorgehensweise in der Praxis jedoch wenig sinnvoll. Der Absender geht häufig von einer Route aus, die ungünstig ist oder bereits nicht mehr funktioniert. Beliebt ist Sourcerouting in Meshnetzwerken deshalb, weil sich Kreisrouten leicht entdecken lassen: Wenn in einer Sourceroutingtabelle ein Knoten mehr als einmal vorkommt, ist offensichtlich etwas faul.

4.1 Funktionsweise des B.A.T.M.A.N.-Algorithmus

Wird der B.A.T.M.A.N.-Routingdaemon in einem Router gestartet, sendet er in einem definierten Intervall per UDP-Broadcast sogenannte *Originatornachrichten* (*Originator-Messages*). Diese sind prinzipiell dasselbe wie Hello-Nachrichten bei Link-State-Protokollen. Der Unterschied zwischen Hellos und Originatornachrichten besteht darin, dass Hello-Nachrichten bei Linkstate-Protokollen nur lokal versendet werden, um Nachbarn in direkter Reichweite zu erkennen. B.A.T.M.A.N. flutet dagegen das ganze Mesh mit Originatornachrichten.

4.1.1 Die Originatornachricht

Eine Originatornachricht enthält die IP-Adresse des Urhebers (Originator), außerdem eine Sequenznummer, mit der die einzelnen Nachrichten vom Urheber fortlaufend nummeriert werden und eine TTL. Der Inhalt der ersten Originatornachricht eines B.A.T.M.A.N.-Nodes mit der Adresse A lautet sinngemäß: „Nachricht von Knoten A, Sequenznummer 0, TTL 50." Andere B.A.T.M.A.N.-Knoten erzeugen und senden ebenfalls eigene Originatornachrichten. Eine Originatornachricht von B.A.T.M.A.N. ist sehr klein, die typische Größe ist 10 Byte. Einschließlich des Overheads, der beim Versenden eines IP-Pakets immer mit anfällt, beträgt die gesamte effektiv übertragene Datenmenge üblicherweise 52 Byte. Im IP-Header eines UDP-Pakets ist immer auch die Adresse des Netzwerkknotens enthalten, welcher das Paket gerade sendet. Diese Information ist für den Protokollablauf wichtig.

B.A.T.M.A.N.-Router wiederholen die Originatornachrichten von anderen Nodes unter der Beachtung bestimmter Regeln – das Netz wird mit Originatornachrichten geflutet. Auf ihrer Reise durch das Mesh gehen die Originatornachrichten dann verloren oder verbreiten sich nur langsam, wenn sie auf ungünstige Bedingungen stoßen (schlechte Funkverbindungen, überlastete Router, viele Interferenzen durch ausgelastete Funkverbindungen). Auf guten, wenig belasteten Funkstrecken kommen sie schnell und mit weniger Verlusten voran.

Empfängt ein B.A.T.M.A.N.-Router ein weitergereichtes Originatorpaket von einem entfernten Knoten, sieht er, von welchem direkten Nachbarn er die

Originatornachricht bekommt. Damit erfährt er zum einen von der Existenz des anderen Knotens und sieht dabei außerdem, hinter welchem Nachbarn sich die Route befindet. Da die Originatornachricht einen Weg gefunden hat, existiert auch eine Route. Gibt es mehrere Routen zu dem entfernten Knoten, kommen die Originatornachrichten auf der besten Route schneller und/oder häufiger an. Liegen die verschiedenen Routen hinter verschiedenen Nachbarn, kann sich der Empfänger einfach für den besten Nachbarn als Gateway entscheiden, wenn er mit dem entfernten Knoten kommunizieren will. Der Empfänger muss sich lediglich merken, von welchem Nachbarn die Originatornachrichten des Senders am schnellsten und am zuverlässigsten eintreffen.

Zu jedem entdeckten Originator führt ein B.A.T.M.A.N.-Router eine Statistik über eine vordefinierte Anzahl der zuletzt empfangenen/erwarteten Originatornachrichten. Ein Node weiß somit, wie viele Originatornachrichten über welchen direkten Nachbarn tatsächlich eingetroffen sind und kann die Qualität der Verbindung beurteilen.

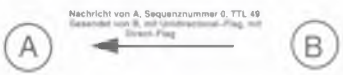
4.1.2 Bidirectional Link Check – Linkprüfung auf Bidirektionalität

Beim Weiterreichen von Originatornachrichten müssen die B.A.T.M.A.N.-Knoten einige einfache Regeln beachten. Dazu gehört, dass nur Originatornachrichten von Nachbarn weitergereicht werden, zu denen eine bidirektionale Kommunikationsverbindung besteht. Eine Route, die in beide Richtungen – auf dem Hin- und Rückweg – funktioniert, kann nur aus einer Kette von Funkverbindungen bestehen, die jeweils bidirektional arbeiten. Außerdem erwartet der Absender beim Transport von Daten, die nicht verbindungslos per Broadcast, sondern per Unicast übertragen werden, eine Empfangsbestätigung (Acknowledgement). Bleibt sie aus, wird die Nachricht mehrfach verschickt. Unidirektionale Funkstrecken sind deshalb für den Transport von Unicast-Paketen unbrauchbar. Da alle Protokolldaten von B.A.T.M.A.N. per Broadcast verschickt werden, muss nachgewiesen werden, dass die Linkstrecken, über die Originatornachrichten empfangen werden, bidirektional sind.

Angenommen, die Knoten A und B sind in einem Mesh direkte Nachbarn, die gegenseitig Daten empfangen können. Knoten B empfängt zum ersten Mal eine Originatornachricht von A. Da die Absenderadresse im IP-Header mit der Originatoradresse bei der empfangenen Originatornachricht identisch ist, weiß B, dass er das Paket direkt von seinem Originator bekommen hat:

B weiß nun, dass A existiert und dass A ein direkter Nachbar ist, dessen Pakete er empfängt. B weiß aber noch nicht, ob A seine Wiederholungen auch empfangen kann. Solange ein Knoten nicht weiß, ob die Verbindung zu einem direkten Nachbarn in beide Richtungen funktioniert (also bidirektional ist), muss er in die Originatornachrichten, die er von diesem Nachbarn wiederholt, eine Warnung hinzufügen, dass die Originatornachricht möglicherweise (oder erwiesenermaßen, falls der Link immer oder meistens unidirektional ist) aus einer einseitigen Kommunikationsbeziehung stammt. Anderenfalls könnten andere Knoten durch den Empfang dieser Aussendung glauben, dass hier eine benutzbare Verbindung besteht, und versuchen, Datenverkehr über die nur in eine Richtung funktionierende Verbindung zu routen.

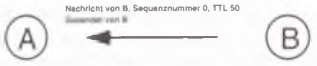
Deshalb fügt B seiner Wiederholung der Originatornachricht von A ein sogenanntes *Unidirectional-Flag* hinzu. B wiederholt nun die Originatornachricht von A, nachdem er die TTL um 1 reduziert hat. Zusätzlich fügt B seinem Antwortpaket noch die Information `Is-Direct-Link` als Flag hinzu, damit A sicher sein kann, dass B auf ein Paket antwortet, dass er direkt von A empfangen hat:



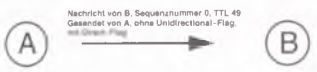
Wenn A die Wiederholung seiner eigenen Nachricht durch B empfängt, weiß er, dass es B gibt und dass B eine direkte Verbindung zu ihm hat. Anhand des gesetzten Flags `Is-Direct-Link` im Originatorpaket erkennt A, dass B seine Originatornachricht direkt von ihm selbst empfangen hat. A weiß nun durch den Empfang seiner eigenen Originatornachricht sicher, dass er einen direkten Nachbarn mit der Adresse B hat, der seine eigenen Originatornachrichten auf direktem Weg empfängt und wieder auf direktem Weg antworten kann. Damit ist aus der Sicht von A die direkte bidirektionale Verbindung zu B nachgewiesen.

Hätte B das Originatorpaket von A nicht empfangen, hätte B es schließlich nicht wiederholt. Hätte B die Nachricht von A über einen Umweg über einen anderen B.A.T.M.A.N.-Knoten empfangen, wäre das `Is-Direct-Flag` nicht gesetzt. Wäre die Originatornachricht auf dem Rückweg verlorengegangen, wüsste A nichts davon, dass B seine Pakete empfängt. A trägt also eine Hostroute zu B in seine Routingtabelle ein. Empfängt jetzt A über B Originatornachrichten von entfernten Originatoren, wiederholt A diese, solange B das beste Gateway zu dem entfernten Originator ist.

A beteiligt sich am Fluten von Nachrichten, die er über B empfängt. Irgendwann schickt B nun ebenfalls aufgrund des festgelegten Intervalls für das Erzeugen von Originatornachrichten eine eigene Originatornachricht auf die Reise:



A weiß beim Empfang dieser Nachricht bereits, dass eine bidirektionale Verbindung zu B besteht und wiederholt diese Nachricht ohne die Unidirectional-Warnung, nachdem er die TTL um 1 reduziert hat. A setzt bei seiner Antwort ebenfalls das `Is-Direct-Link`-Flag:



B empfängt daraufhin die Wiederholung seiner eigenen Originatornachricht von A. Endlich wissen A und B, dass sie direkte Nachbarn sind, die sich gegenseitig sehen und in beide Richtungen miteinander kommunizieren können. Auch B trägt nun eine Hostroute zu A in seine Routingtabelle ein. Empfängt B Originatornachrichten von entfernten Originatoren, die von A weitergereicht werden, wiederholt B diese nun auch.

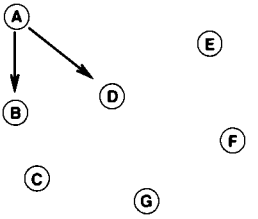
Angenommen, hinter Knoten B existiert noch der Knoten C. B und C haben eine bidirektionale Verbindung und beide wissen das auch schon, da sie den Bidirectional Link Check bereits erfolgreich durchlaufen haben. Wenn B die Originatornachrichten von A wiederholt, hört C die Übertragungen ebenfalls. So erfährt C, dass sich A hinter B befindet. C trägt B als Gateway zu A in seine Routingtabelle ein. Wiederholt B aus der anderen Richtung die Originatornachrichten von C, erfährt A von der Existenz von C. A trägt B als Gateway zu C ein.

Da beide Funkverbindungen A–B und B–C daraufhin geprüft wurden, dass sie in beide Richtungen funktionieren, gibt es nun eine funktionierende Route A–B–C und eine funktionierende Route C–B–A. Über den Verlauf der beiden Routen wissen A und C jedoch nichts – sie wissen lediglich, dass B für sie das nächste Gateway ist. Wüsste A, dass C seine Originatornachrichten mit einer TTL von 50 versieht, könnte A sich wegen der TTL von 49 in den Originatornachrichten, die er von B empfängt, ausrechnen, dass C sich unmittelbar hinter B befindet. Eine Festlegung auf eine bestimmte TTL existiert in B.A.T.M.A.N. jedoch nicht. Damit kann jeder Knoten entscheiden, wie weit (im Sinne von Hops) er seine Anwesenheit dem Netz mitteilen möchte.

4.1.3 Eine Originatornachricht reist durch das Mesh

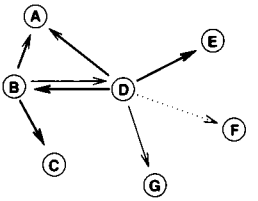
Die folgenden Grafiken verfolgen das Fluten eines kleinen Meshnetzes mit einer Originatornachricht der Station A. Die Qualität der Funkstrecken ist durch unterschiedlich kräftige Pfeile angedeutet.

Im ersten Schritt sendet A eine selbst erzeugte Originatornachricht, diese wird empfangen von B und D. Diese prüfen anhand der Urheberadresse und der Sequenznummer, ob sie diese Nachricht schon einmal gesehen haben. Wird die Nachricht zum ersten Mal empfangen, wiederholen sie die Ausstrahlung, nachdem sie die TTL reduziert haben:



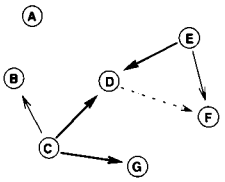
B und D sehen die Nachricht zum ersten Mal und wiederholen sie. A sieht, dass B und D seine eigene Nachricht wiederholen. In diesem Beispiel ist die Bidirektionalitätsprüfung zwischen A, B und D bereits erfolgreich abgeschlossen. Daher wiederholen B und D die Originatornachricht von A ohne das Unidirectional-Flag. Nun sieht auch C die von B wiederholte Nachricht. Die Funkverbindung von B zu D funktioniert schlechter als von D zu B, sie ist gelegentlich unidirektional.

In unserem Beispiel geht das Paket auf dem Weg von B zu D verloren. Die Wiederholung durch D wird auch von E und B gesehen. Anhand der Sequenznummer erkennt B, dass die von D wiederholte Nachricht bereits bekannt ist und ignoriert sie. Die Übertragung von D zu G gelingt manchmal, in diesem Augenblick ist sie jedoch nicht erfolgreich:

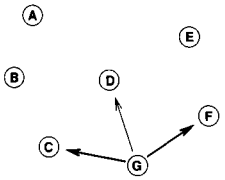


Nun wiederholen C und E die Nachricht. Die Verbindung von E zu F ist unzuverlässig, diesmal geht die Originatornachricht auf dem Weg von E zu F verloren. D sieht die Wiederholung der Nachricht von E, die bereits bekannt ist und somit ignoriert wird. Auch B und D ignorieren die Wiederholung von C. Sporadisch kommen Nachrichten von D bei F an, in der Beispielsituati-

on ist das der Fall. Der Link funktioniert aber nur von D nach F. Da F von D die Nachricht über einen unidirektionalen Link bekommt, wird sie von F ignoriert. Weitergereichte Originatornachrichten werden von B.A.T.M.A.N. nur beachtet, wenn sie von einem Nachbarn empfangen werden, der über eine bidirektionale Verbindung erreicht werden kann:



Bei der nächsten Übertragung kommt die Originatornachricht von A über G bei F an. C und D ignorieren die Wiederholung der bekannten Originatornachricht:



Nachrichten von A können über mehrere Wege bei F ankommen. Möglich sind unter anderem die Routen A–B–C–G–F, A–D–E–F, A–B–D–E–F. Wichtig für F ist nur, dass die Originatornachrichten von A entweder über G oder E zuerst bei ihm ankommen. Wenn Node F nun mit A kommunizieren möchte, muss er sich lediglich entscheiden, ob er G oder E mit dem Weiterreichen der IP-Pakete beauftragt. Diese Entscheidung kann F treffen, indem er eine Statistik führt, über welchen direkten Nachbarn wieviele Originatornachrichten von A zuerst eingetroffen sind (Tabelle 4.1).

Tabelle 4.1: Ranking-Tabelle

Stationen	Über welchen Nachbarn gesehen?	
	G	E
A	14	7
B	16	4
C	18	8
D	14	9

Fortsetzung:

Stationen	Über welchen Nachbarn gesehen?	
E	22	19
G	59	0

In der Beispielsituation ist die Route A–B–C–G–F zuverlässiger aber langsamer als beispielsweise A–D–E–F. Die kürzere Route kann zwar schneller sein, aber wenn die Mehrzahl der Pakete auf dieser Route nicht durchkommt, werden die über G eintreffenden Pakete häufiger das Rennen gewinnen.

4.2 Praxis mit B.A.T.M.A.N.

Der B.A.T.M.A.N.-Daemon steht bislang für Linux, FreeBSD und Macintosh OS-X zur Verfügung. Die Entwicklungsarbeit konzentriert sich jedoch in erster Linie auf Linux, weshalb es vorkommen kann, dass erweiterte Funktionen unter anderen Betriebssystemen erst mit einer gewissen Verzögerung zur Verfügung stehen.

Linux-Installationspakete des B.A.T.M.A.N.-Daemon `batmand` gibt es für Debian, OpenZaurus und OpenWRT.[1] Zum Kompilieren aus dem Quelltext genügt ein einfaches `make` und `make install` im Sourcecodeverzeichnis. Als einzige Abhängigkeit wird die Bibliothek `libpthread` vorausgesetzt, die auf einem Linux-System üblicherweise bereits installiert sein sollte.

Um über ein B.A.T.M.A.N.-Mesh ins Internet gehen zu können, muss außerdem unter Linux das Kernelmodul `tun` installiert sein. Es ist im Standardkernel der Linuxdistributionen enthalten und wird beim ersten Start des B.A.T.M.A.N.-Daemons automatisch geladen. Wer einen selbstkompilierten Kernel einsetzt, findet es zum Beispiel in `xconfig` in der Abteilung `Network Device Support` unter der Bezeichnung `Universal TUN/TAP device driver support`.

B.A.T.M.A.N. handelt automatisch UDP-Tunnel zu Internet-Gateways aus. IP-Pakete, die ins Internet gesendet werden, werden in UDP-Pakete verpackt und vom Gatewayclient an das Gateway geschickt. Auch wenn die IP-Pakete im Tunnel über mehrere Hops im Mesh transportiert werden, sieht der Tunnel für sie wie eine direkte Verbindung (Single-Hop) zum Gateway aus. Anfangs- und Endpunkt des Tunnels sind festgelegt und unabhängig davon, wie die Route im Mesh dazwischen gerade verläuft. Solange das Mesh in der Lage ist, eine funktionierende Hin- und Rückroute zwischen Client und Gateway bereitzustellen, ist die Anbindung an das Internet stabil, auch wenn sich der Routingpfad zwischen beiden Endpunkten häufig verändert.

[1] `http://downloads.open-mesh.net`

Durch die Verwendung eines Tunnels kann der Internet- oder Gatewaynutzer auf der Clientseite das Gateway aussuchen und bestimmen, dass das Gateway nicht umgeschaltet wird. Falls ein Gateway für Internetverkehr zum schwarzen Loch wird oder nur noch langsam reagiert, kann der Client selbst das Gateway abwählen und einen neuen bestimmen. Außerdem kann der Client entscheiden, wann, unter welchen Umständen und nach welchen Kriterien ein Gateway ausgesucht beziehungsweise gewechselt wird.

Hat der B.A.T.M.A.N.-Daemon erfolgreich einen Tunnel aufgebaut, legt er ein Tunnelinterface `tun0` an. Bei einer soliden Linuxdistribution sollten beim Einsatz des Kernelmoduls `tun` keine Probleme auftreten. Manchmal kommt es vor, dass der Devicenode `/dev/net/tun` nicht vorhanden ist oder nicht automatisch angelegt wurde. In diesem Fall kann der Deviceknoten von Hand mit Administratorrechten erzeugt werden:

```
mknod -m 600 /dev/net/tun c 10 200
```

4.2.1 Erster Start des Daemons

B.A.T.M.A.N. wird mit Administratorrechten von der Kommandozeile mit allen erforderlichen Parametern gestartet, eine Konfigurationsdatei gibt es nicht. Wie bei den meisten Kommandozeilenprogrammen gibt `batmand -h` einen kurzen Hilfetext aus. Ausführlichere Informationen erhält man mit der Option `-H`

Es können mehrere Interfaces angegeben werden, die natürlich funktionsfähig und konfiguriert sein müssen. Im einfachsten Fall genügt dann der Aufruf

```
batmand Interface1 Interface2 .. InterfaceN
```

Dabei wird das Default-Originatorintervall von einer Originatornachricht pro Sekunde und pro Interface verwendet. Der Routing-Algorithmus lässt sich über die Einstellung des Originatorintervalls an unterschiedliche Einsatzzwecke anpassen.

Wenn in einem Mesh schnelle Reaktionen auf Veränderungen der Topologie gewünscht sind und dafür etwas mehr Protokolloverhead in Kauf genommen werden kann, kann das Originatorintervall relativ klein gesetzt werden. Ein typischer Anwendungsfall sind kleinere Netze mit vielen mobilen Knoten.

Wenn geringer Protokolloverhead im Vordergrund steht, kann der Wert vergrößert werden. Ein sehr großes Mesh mit vielen hundert Knoten würde unter einem zu kleinen Originatorintervall leiden, hier fordert die große Originatoranzahl Abstriche bei der Reaktionsgeschwindigkeit.

Ist ein kürzeres oder längeres Originatorintervall erforderlich, kann das mit der Option `-o` angegeben werden, die Eingabe erfolgt in Millisekunden:

```
batmand -o 2000 Interface1 Interface2 ... InterfaceN
```

4.2.2 Den Netzwerkstatus im Debugging-Modus beobachten

Der B.A.T.M.A.N.-Daemon kann auch im Debugging-Modus mit der Option `-d` und einem Debuglevel von 0 bis 4 gestartet werden, um den Status des Netzwerks zu beobachten:

```
batmand -d Debuglevel Interface
```

Ohne die Debugging-Option oder mit der Angabe des Debuglevels 0 geht `batmand` beim Start sofort in den Hintergrund und routet auf den angegebenen Interfaces.

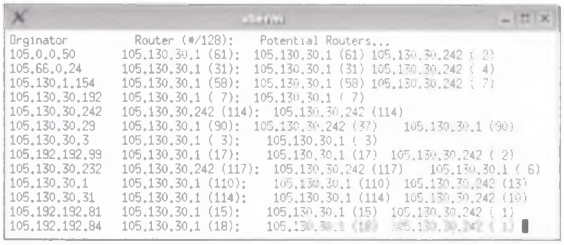


Abbildung 4.9: Ausgabe im Debuglevel 1

Wird das Programm in einem Debuglevel größer als Null aufgerufen, bleibt der Daemon im Vordergrund und gibt auf dem Terminal die somit angeforderten Informationen aus. Für Anwender sind nur die Debuglevel 1 und 2 interessant. Debuglevel 1 zeigt eine Liste aller erreichbaren Originatoren im Mesh an, außerdem die Nachbarn, über die zu den Originatoren geroutet werden kann und die Anzahl der jeweils empfangenen Originatornachrichten. Existieren Routen über mehrere Nachbarn zu einem Originator, werden diese in der Liste der `Potential Routers` angezeigt (Abbildung 4.9).

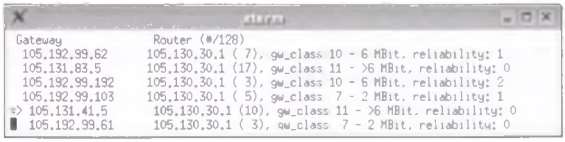


Abbildung 4.10: Ausgabe von Debuglevel 2

Mit der Debug-Option `-d 2` werden im Connect-Mode die vorhandenen Gateways aufgelistet. Abbildung 4.10 zeigt ein Beispiel, bei dem das Gateway mit der IP 105.131.41.5 ausgewählt wurde. Es offeriert dem Mesh einen Internet-Uplink mit mehr als 6 MBit. Von den 128 Originatornachrichten, die das Gateway auf die Reise geschickt hat, sind allerdings nur 10 auf dem lokalen Rechner angekommen, weshalb der erzielbare Datendurchsatz gering sein dürfte. Der Wert `Reliability` zeigt an, wie zuverlässig die Verbindung zum Gateway ist. Gateway und Gateway-Client kommunizieren regelmäßig darüber, ob der Tunnel aufrechterhalten werden soll. Schlägt diese Kommunikation fehl, erhöht sich der Reliability-Wert.

Der Verlauf der Route zu diesem Gateway kann unter Linux und BSD mit dem Befehl `ping -R` *`Ziel-IP`* überprüft werden. Dabei wird der Hin- und Rückweg angezeigt:

```
linux:~ #  ping -R 105.131.41.5
PING 105.131.41.5 (105.131.41.5) 56(124) bytes of data.
64 bytes from 105.131.41.5: icmp_seq=1 ttl=60 time=37.7 ms
RR:     105.130.30.0
        105.130.30.1
        105.130.1.67
        105.131.83.3
        105.131.41.5
        105.131.41.5
        105.131.41.3
        105.130.1.67
        105.130.30.1

64 bytes from 105.131.41.5: icmp_seq=2 ttl=60 time=10.2 ms  (same route)
```

Die Anzeige der Ping-Ergebnisse erfolgt fortlaufend. Ändert sich die Route, ist das sofort zu erkennen. Das RR in der Ausgabe steht für *Record Route*. Zu jedem erreichbaren Knoten kann so die Route geprüft werden, sofern Ping nicht durch Firewalleinstellungen blockiert wird.

Debugging im Connect-Modus

Hat man den B.A.T.M.A.N.-Daemon bereits ohne das gewünschte Debuglevel gestartet oder möchte gleichzeitig oder abwechselnd die Ausgaben der Debuglevels 1 und 2 sehen, hat man die Möglichkeit, jederzeit beliebig viele Instanzen des B.A.T.M.A.N.-Daemon im Connect-Modus zu starten. Dazu gibt man den Schalter `-c` zusammen mit dem Schalter `-d` *`1-4`* für das Debuglevel an. Läuft ein mit dem Routing beschäftigter B.A.T.M.A.N.-Daemon im Hintergrund, kann man sich mit

```
batmand -c -d 1
```

den Routingstatus der routenden Instanz des batmand ansehen. Dabei wird die Anzeige fortlaufend aktualisiert. Soll der batmand nur einmalig eine Ausgabe erzeugen und sich dann beenden, kann man ihn mit der Option -b im Batch-Mode starten. Das ist interessant für Skripte, welche die Ausgabe des Daemons verarbeiten:

```
batmand -c -b -d 1
```

4.2.3 Routingklassen und Gatewayklassen

Der Daemon kennt unterschiedliche Gatewayklassen für Internet-Gateways, das heißt, die Gateways geben im Mesh bekannt, wieviel Bandbreite sie brutto zum Internet anbieten.

Gateway-Konfiguration

Ein B.A.T.M.A.N.-Gateway, das einen Internetzugang mit 2 MBit Bandbreite (Klasse 7) anbietet, startet man wie folgt:

```
batmand -g 7 eth1 wlan0
```

Auch das Annoncieren einer Route in ein anderes Netzwerk, Subnetz oder Hostroute analog zu den HNA-Announcements von OLSR ist möglich:

```
batmand -a 10.145.145.0/24 eth1 wlan0
```

Selbstverständlich kann man beide Optionen kombinieren und auch mehrere Host Network Announcements gleichzeitig vornehmen:

```
batmand -g 7 -a 10.11.12.13/32 10.22.33.0/22 eth1 wlan0
```

Gateway-Clients

Wird ein B.A.T.M.A.N.-Daemon gestartet, sucht er nicht automatisch eine Route zum Internet. Will man auf Clientseite angeben, dass er ins Internet routen soll, kann man mit der Option -p ein bevorzugtes Gateway angeben. Zusätzlich muss mit -r eine Routingklasse angegeben werden. Falls das bevorzugte Gateway nicht erreichbar ist, wird sich der B.A.T.M.A.N.-Daemon ein alternatives Gateway nach den vorgegebenen Kriterien der Routingklasse aussuchen. Drei Routingklassen werden angeboten.

Klasse 1 ist auf maximale Bandbreite zum Internet optimiert. Um das zu erreichen, sucht sich der B.A.T.M.A.N.-Daemon den besten Kompromiss

aus der vom Gateway angegebenen Gatewayklasse und der Verbindungsqualität über das Mesh zum Internet-Gateway. Bei Klasse 1 gibt er einem schnelleren Gateway den Vorzug, auch wenn dann die Verbindung weniger zuverlässig sein sollte:

```
batmand -r 1  ath0
```

Klasse 2 ist auf maximale Verbindungsqualität zum Internet-Gateway ausgerichtet. Der Routing-Daemon wird das am besten erreichbare Gateway wählen, auch wenn dieses nur langsamen Internetzugang anbietet. Das ist dann interessant, wenn eine langsame, dafür jedoch zuverlässige Internetverbindung Priorität haben soll, z. B. beim Chat, bei SSH-Verbindungen und VOIP-Anwendungen:

```
batmand -p 105.131.41.5 -r 2  ath0
```

Klasse 3 wählt automatisch das jeweils nächstgelegene Internet-Gateway. Diese Einstellung ist für mobile Knoten interessant, die sich viel im Mesh bewegen. Eine solche Verbindung ist schnell hergestellt, bedingt aber häufiges Umschalten des Gateways und führt deshalb oft zu Verbindungsabbrüchen bei Anwendungen, die nicht verbindungslos sind. Klasse 3 entspricht der Gatewayfunktionalität von OLSR mit den bekannten Problemen (Gateway-Switching, Verbindungsabbrüche) und Vorteilen (schneller Verbindungsaufbau).

Hat der Tunnelaufbau geklappt, findet man bei einem Blick in die Routingtabelle eine Defaultroute (Ziel: 0.0.0.0) über ein Interface mit dem Namen `tun`*`Nummer`*.

Die im Mesh verfügbaren Knoten sind jeweils mit einer Hostroute eingetragen. Hostrouten sind erkennbar an der Netzmaske 255.255.255.255. Die IP-Routingtabelle kann entweder mit dem Befehl `netstat -rn` oder mit `ip r` abgefragt werden. Auf Embedded-Systemen (OpenWRT) findet man meist das modernere `iproute`-Paket vor, auf Arbeitsplatzrechnern steht üblicherweise `netstat` zur Verfügung. Der Befehl `ip r` funktioniert nur auf Systemen, die das Paket `iproute` (oder `iproute2`) installiert haben. Die Ausgaben beider Programme sehen unterschiedlich aus, sind aber inhaltlich gleich:

```
linux:~ #  netstat -rn

Kernel IP Routentabelle
Ziel            Router          Genmask         Flags  MSS Fenster irtt Iface
105.130.77.80   105.130.30.1    255.255.255.255 UGH      0 0          0 eth2
105.130.1.67    105.130.30.1    255.255.255.255 UGH      0 0          0 eth2
[...]
105.130.30.1    0.0.0.0         255.255.255.255 UH       0 0          0 eth2
```

```
105.192.99.192 105.130.30.1  255.255.255.255 UGH      0 0          0 eth2
105.192.99.0   105.130.30.1  255.255.255.192 UG       0 0          0 eth2
105.0.0.0      0.0.0.0       255.0.0.0       U        0 0          0 eth2
0.0.0.0        0.0.0.0       0.0.0.0         U        0 0          0 tun0
```

```
linux:- # ip r

105.130.77.80 via 105.130.30.1 dev eth2
105.130.1.67 via 105.130.30.1 dev eth2
105.131.41.1 via 105.130.30.1 dev eth2
(...)
105.130.30.1 dev eth2  scope link
105.192.99.192 via 105.130.30.1 dev eth2
105.192.99.0/26 via 105.130.30.1 dev eth2
105.0.0.0/8 dev eth2  proto kernel  scope link  src 105.130.30.38
default dev tun0  scope link
```

Eine Installation des Kernelmoduls `tun` ist nur dann unnötig, wenn ein B.A.T.M.A.N.-Knoten lediglich als Meshrelais arbeitet, selbst also keine Verbindung zu einem Internet-Gateway aufbaut und auch kein Internet-Gateway anbietet.

4.2.4 Visualisierungsserver

Visualisierungen von Meshnetzwerken sind beliebt und in der Praxis wegen der schnellen Übersicht nützlich. Doch bedingt durch das Konzept des B.A.T.M.A.N.-Algorithmus kennen die einzelnen Meshknoten die Gesamttopologie des Meshnetzwerks nicht. Deshalb haben sich die Entwickler eine Lösung ausgedacht, die es ermöglicht, Topologieinformationen über die Gesamttopologie des B.A.T.M.A.N.-Mesh zu ermitteln. Das Fluten des gesamten Netzwerks mit Topologieinformationen zum Zwecke der Visualisierung und das Erstellen einer Topologiedatenbank in jedem einzelnen Meshknoten wie bei einem Link-State-Protokoll würde das Gesamtkonzept von B.A.T.M.A.N. jedoch ad absurdum führen und den proaktiven Overhead wieder durch die Hintertür hereinbringen.

Statt die Topologieinformationen über das Mesh zu fluten und in jedem Knoten zu sammeln, kann ein leistungsfähiger Visualisierungsserver bei jedem routenden Daemon (egal ob Gateway-Client oder Gateway) mit der Option `-s` angegeben werden. Die einzelnen Knoten schicken ihre Informationen an den Visualisierungsserver. Der Server sammelt die Topologieinformationen und gibt sie in einem zu Graphviz kompatiblen Format aus, so dass die für Olsrd zur Verfügung stehenden Visualisierungslösungen auch mit B.A.T.M.A.N. funktionieren.

Das folgende Beispiel ruft `batmand` als Gateway-Client von Gateway 105.131.41.5 mit der Routingklasse 2, aus, sorgt dafür, dass die erreichbaren Ori-

ginatoren und Router per Debuggingausgabe sichtbar werden und wählt 105.192.192.230 als Visualisierungsserver aus:

```
batmand -d 1 -p 105.131.41.5 -r 2 -s 105.192.192.230 ath0
```

Es muss kein Gateway angegeben werden, wenn dieses nicht explizit festgelegt werden soll. Die Angabe einer Routingklasse reicht aus. Umgekehrt muss aber bei der Angabe eines bevorzugten Gateways eine Routingklasse angegeben werden. Ist dieses Gateway nicht erreichbar, wählt der Daemon entsprechend der Routingklasse eine Alternative.

4.2.5 Firewalleinstellungen anpassen

`batmand` verwendet die Ports 1966 (UDP) für Protokollnachrichten und Port 1967 (UDP und TCP) für den Transport der IP-Pakete im UDP-Tunnel. Diese beiden Ports müssen für eingehenden und ausgehenden Datenverkehr durch die Firewall geöffnet sein. Die Portnummern können sich nach Erscheinen dieses Buches im Zuge der Standardisierung des B.A.T.M.A.N.-Protokolls und der offiziellen Vergabe durch die IANA (Internet Assigned Numbers Authority) noch ändern.

In Computern, auf denen der B.A.T.M.A.N.-Visualisierungsserver installiert wird, muss der UDP-Port 1968 geöffnet sein, um Topologie-Nachrichten empfangen zu können.

Wenn über einen B.A.T.M.A.N.-Gateway-Client lokale Computer (die beispielsweise am LAN-Port des Routers angeschlossen sind) ins Internet geroutet werden sollen, muss zum UDP-Tunnel hin NAT eingerichtet werden. Das folgende Beispiel ist betont einfach gehalten, und löscht im ersten Schritt mit dem Befehl `iptables -F` alle eventuell bereits vorhandenen Firewall-Regeln. 105.130.30.16 ist die IP-Adresse des Tunnelinterfaces am B.A.T.M.A.N.-Gateway-Client:

```
#!/bin/sh
iptables -F; iptables -t nat -F; iptables -t mangle -F
iptables -t nat -A POSTROUTING -o tun0 -j SNAT --to 105.130.30.16
```

Auf dem B.A.T.M.A.N.-Gateway muss ebenfalls zum Internet hin NAT eingerichtet werden, wie im Abschnitt 3.5.2 ab Seite 69 beschrieben.

4.2.6 Webinterface für B.A.T.M.A.N.

Für die Freifunk-Firmware gibt es ein Webinterface von Ludger Schmudde, mit dem der `batmand` mittels Webbrowser komfortabel konfiguriert und seine Funktion beobachtet werden kann. Die Installation wird im Abschnitt 6.5.1 ab Seite 156 beschrieben.

Kapitel 5

WLAN-Technik

Dieses Kapitel widmet sich der für Mesh-Netze benötigten Technik. Wer begierig darauf sind, einen Mesh-Router sofort auszuprobieren und einzurichten, kann es vorerst überspringen und später lesen – gesetzt den Fall, man entscheidet sich für eines der im Kapitel 6 ab Seite 125 vorgestellten SoHO[1]-Router-Modelle.

Lesen sollten man dieses Kapitel aber auf jeden Fall, denn hier steht vieles, was für einen langfristigen Betrieb eines Mesh-Routers nötig ist, von den WiFi-Spezifikationen über die Eigenheiten spezieller Chipsätze bis hin zu zahlreichen praktischen Tipps für die Praxis, etwa zur Verkabelung, Stromversorgung oder dem Langfrist-Betrieb von Routern im Freien. Das Kapitel ist auch für all jene wichtig, die einen PC oder Laptop als Mesh-Router betreiben wollen.

[1] Die Abkürzung steht für *Small or Home Office*.

5.1 Die WiFi-Standards 802.11 von a bis Draft-n

Etabliert sind WiFi-Geräte nach den Standards des Institute of Electrical and Electronics Engineers (IEEE) 802.11b, 802.11g und 802.11a. Der IEEE-Standard 802.11n MIMO (Multiple Input, Multiple Output) ist zwar noch nicht fertiggestellt, aber es sind bereits Produkte im Handel, die sich als 802.11 Draft-n oder Pre-n bezeichnen.

5.1.1 IEEE 802.11b/g

Die zwei am weitesten verbreiteten Standards IEEE 802.11b und 802.11g funken in Deutschland, Österreich und der Schweiz auf 13 Kanälen im sogenannten ISM-Frequenzband bei 2,4 Gigahertz. ISM steht für „Industry, Science, Medical" und bezeichnet verschiedene Bereiche des elektromagnetischen Frequenzspektrums, in dem unterschiedliche Anwendungen lizenzfrei arbeiten dürfen. Der ISM-Frequenzbereich wird dabei nicht nur für Kommunikationstechnologie genutzt, sondern auch für Anwendungen, bei denen beispielsweise mit sehr hohen Sendeleistungen Hitze erzeugt wird.

Unter anderem arbeiten Mikrowellenöfen in Privathaushalten im gleichen Frequenzbereich und senden mit einigen hundert Watt innerhalb des Gehäuses. Dabei tritt immer auch ein Teil der Sendeenergie nach außen. Im Vergleich dazu ist die maximale effektive Sendeleistung von 802.11b/g in Europa mit 0,1 Watt sehr gering. Die aus dem Gehäuse entweichende Störstrahlung eines Mikrowellenherds kann vor allem bei älteren oder defekten Geräten leicht ein Vielfaches der Sendeenergie eines Funknetzes betragen, weshalb von Mikrowellenherden verursachte Störungen in WLANs häufig sind.

Der 2,4-GHz-Frequenzbereich wird wegen der vielen sich gegenseitig störenden Anwendungen mitunter abwertend als „Schrottband" bezeichnet. In einem kleinen Abschnitt des elektromagnetischen Spektrums von nur 60 Megahertz Breite drängeln und behindern sich viele konkurrierende Anwendungen wie beispielsweise RFID-Lesegeräte, Bluetooth, drahtlose Headsets, Funkmäuse, Funktastaturen, Videosender und auch der Amateurfunk. Besonders häufig werden WLANs von analogen Videoübertragungssystemen gestört, die ein Dauersignal senden.

Tabelle 5.1: Kanäle im 2,4-GHz-Bereich

Kanal	Frequenz	verwendet in
01	2.412 GHz	EU, Japan, USA
02	2.417 GHz	EU, Japan, USA
03	2.422 GHz	EU, Japan, USA
04	2.427 GHz	EU, Japan, USA
05	2.432 GHz	EU, Japan, USA

Fortsetzung:

Kanal	Frequenz	verwendet in
06	2.437 GHz	EU, Japan, USA
07	2.442 GHz	EU, Japan, USA
08	2.447 GHz	EU, Japan, USA
09	2.452 GHz	EU, Japan, USA
10	2.457 GHz	EU, Japan, USA
11	2.462 GHz	EU, Japan, USA
12	2.467 GHz	EU, Japan
13	2.472 GHz	EU, Japan
14	2.484 GHz	Japan

Das *Kanalraster*, also der Abstand zwischen zwei Kanalmittelpunkten, beträgt in Amerika und Europa 5 MHz, die tatsächlich verwendete Bandbreite bei der Datenübertragung jedoch +/- 11 MHz, insgesamt also 22 MHz. Durch die hohe Bandbreite überlappen sich viele Kanäle. Benachbarte WLAN-Funkzellen auf nahe zusammenliegenden Kanälen stören sich gegenseitig, indem sie Interferenzen verursachen, die zu Verlusten bei der Reichweite und der erzielbaren Übertragungsgeschwindigkeit führen. Günstig wäre es, wenn sich die verschiedenen WLANs auf die Kanäle 1, 5, 9 und 13 verteilen würden – damit würden vier Kanäle zur Verfügung stehen, die sich kaum überlappen.

In Europa verkaufte Geräte werden nicht selten mit einer Firmware für die USA ausgeliefert und gestatten nur den Betrieb auf den US-Kanälen 1 – 11. Als einziges Land gestattet Japan auch die Nutzung von Kanal 14, der wegen seines Kanalabstands von 12 Megahertz zu Kanal 13 etwas aus dem Rahmen fällt.

Der Betrieb eines Netzwerks auf Kanal 13 ist naheliegend, da wenige Netze auf diesem Kanal arbeiten. Die meisten Accesspoints senden auch in Europa bevorzugt auf den Kanälen 1, 6 und 11, da sich diese Kanäle nicht überlappen und keine Probleme bei der Kanalwahl mit amerikanischen Firmwareversionen auftreten. Wer der dichten Belegung auf den unteren 11 Kanälen ausweichen will, sollte beim Kauf darauf achten, dass alle 13 Kanäle funktionieren.

802.11b und 802.11g unterscheiden sich lediglich in der erreichbaren Geschwindigkeit und der bei der Datenübertragung verwendeten Modulationsarten. 802.11b kann – abhängig von der Übertragungsqualität (dem Verhältnis des Nutzsignals zum Rauschen, Signal-to-Noise-Ratio SNR) – mit Bruttodatenraten von 11, 5,5, 2 und 1 Megabit arbeiten. 802.11g setzt auf Bruttodatenraten von 54, 48, 36, 24, 18, 12 und 6 Megabit. 802.11g ist abwärtskompatibel zu 802.11b, solange der Kompatibilitätsmodus nicht abgeschaltet ist.

Die höheren Geschwindigkeiten von 802.11g erfordern ein besseres Signal-Rauschverhältnis gegenüber 802.11b. Eine höhere Datenrate hat deshalb eine geringere Reichweite zur Folge und umgekehrt. Auf Funkstrecken, die die normale WLAN-Reichweite überschreiten, ist 802.11g nur dann schneller als 802.11b, wenn das Signal-Rauschverhältnis für die schnelleren Datenraten von 802.11g ausreichend ist. Tatsächlich arbeiten viele WLAN-Karten bei gleichem Signal-Rauschabstand im b-Modus noch mit 11 Megabit, während im g-Modus die Datenrate bereits auf 6 Megabit reduziert wird. 802.11b gehört daher bei Langstreckenverbindungen im 2.4 GHz-Band noch nicht zum alten Eisen. Doch die Frage, welchen Standard neue Hardware für das 2,4 GHz-Band unterstützen soll, stellt sich heute kaum noch, da Geräte nach 802.11b aus dem Handel praktisch verschwunden sind.

Einige Hersteller bewerben für ihre Produkte Bruttodatenraten, die nicht konform zum Standard sind, wie etwa Super-G, 802.11b+ oder Turbo-Mode mit 108 Megabit. Diese Verfahren sind proprietär, funktionieren nur auf sehr kurze Entfernungen und nur mit Karten desselben Chip-Herstellers. Für den praktischen Betrieb in einem Mesh sind sie deshalb kaum von Bedeutung.

5.1.2 IEEE 802.11a

802.11a unterscheidet sich von 802.11b und 802.11g vor allem durch den verwendeten Frequenzbereich zwischen 4,9 und 5,9 GHz. Dieser Bereich des elektromagnetischen Spektrums ist im Gegensatz zu 2,4 GHz nicht von zahllosen Anwendungen überlaufen. Die verfügbaren Kanäle weichen von Staat zu Staat voneinander ab und der Einsatz von 802.11a ist in weitaus weniger Ländern freigegeben als bei 802.11b/g. In Europa werden zwei Frequenzbereiche von 5,150 GHz bis 5,350 GHz und von 5,470 GHz bis 5,725 GHz verwendet. Der Betrieb im Freien ist nur im oberen Frequenzband zugelassen. Im unteren Frequenzband ist eine maximale effektive Sendeleistung von 0,2 Watt erlaubt, im oberen Frequenzband dürfen es bis zu 1 Watt sein. 802.11a konkurriert im unteren Frequenzband nur mit High Performance Radio Local Area Network (HIPERLAN), einem weiteren, weniger verbreiteten Standard für drahtlose Computernetze. Störungen von anderen Anwendungen sind in diesem Frequenzbereich kaum zu erwarten.

Das obere Frequenzband teilt sich 802.11a mit Radarsystemen. Dabei wird selbstverständlich dem Radar Priorität eingeräumt. Geräte für 802.11a müssen mit einer Technologie (802.11h) ausgestattet sein, die Kollisionen mit Radar verhindert und gegebenenfalls automatisch einen Frequenzwechsel durchführt. Da ein automatischer Frequenzwechsel im Ad-Hoc-Modus technisch schwierig zu organisieren und in 802.11a auch nicht vorgesehen ist, ist diese Betriebsart im 5-GHz-Bereich in Deutschland nicht gestattet. Aus diesem Grund kann 802.11a nicht legal in einem Mesh für Multipoint-zu-Multipoint-Verbindungen eingesetzt werden. Es lassen sich aber perfor-

mante Punkt-zu-Punkt-Verbindungen aufbauen, über die ein Mesh Traffic routen kann. Die zur Verfügung stehenden Bruttodatenraten sind die gleichen wie bei 802.11g. 802.11a ist bislang viel weniger verbreitet als 802.11b und 802.11g. Bei Elektronikdiscountern findet man Geräte für 802.11a eher selten im Angebot. Für lange Funkstrecken ist 802.11a interessant, da im Outdoor-Bereich eine Sendeleistung bis zu 1 Watt erlaubt ist und Störungen von anderen Funknetzen wegen der geringen Verbreitung seltener auftreten. Die Übertragungsbandbreite beträgt bei 802.11a 20 MHz, die verwendeten Kanäle überlappen sich nicht.

Die Tabellen 5.2 und 5.3 zeigen die Kanäle im 5-GHz-Bereich, nach Zulassungen im Gebäude oder im Freien getrennt.

Kanal	Frequenz	verwendet in
36	5,180 GHz	EU, USA, Japan
40	5,200 GHz	EU, USA, Japan
44	5,220 GHz	EU, USA, Japan
48	5,240 GHz	EU, USA, Japan
52	5,260 GHz	EU, USA
56	5,280 GHz	EU, USA
36	5,180 GHz	EU, USA
40	5,200 GHz	EU, USA
44	5,220 GHz	EU, USA
48	5,240 GHz	EU, USA
52	5,260 GHz	EU, USA
56	5,280 GHz	EU, USA
60	5,300 GHz	EU, USA
64	5,320 GHz	EU, USA

Tabelle 5.2: Kanäle im 5-GHz-Bereich (nur in Gebäuden)

Kanal	Frequenz	verwendet in
100	5,500 GHz	EU
104	5,520 GHz	EU
108	5,540 GHz	EU
112	5,560 GHz	EU
116	5,580 GHz	EU
120	5,600 GHz	EU
124	5,620 GHz	EU

Tabelle 5.3: Kanäle im 5-GHz-Bereich (auch im Freien)

Fortsetzung:

Kanal	Frequenz	verwendet in
128	5,640 GHz	EU
132	5,660 GHz	EU
136	5,680 GHz	EU
140	5,700 GHz	EU
147	5,735 GHz	USA
151	5,755 GHz	USA
155	5,775 GHz	USA
167	5,835 GHz	USA

5.1.3 IEEE 802.11n

IEEE 802.11n, auch als MIMO (Multiple Input, Multiple Output) bezeichnet, bildet die Grundlage für die nächste Generation drahtloser WLAN-Geräte. Neben höherer Datenraten bis zu 300 Mbps sollen Geräte nach diesem Standard eine größere Reichweite bei geringerem Stromverbrauch erzielen. Erreicht wird die größere Reichweite und Bandbreite durch mehrere Antennen, Sender und Empfänger in einer WLAN-Karte. Der Standard ist noch nicht offiziell verabschiedet, es existiert jedoch seit 2006 ein vorläufiger Entwurf. Unter der Bezeichnung 802.11 *Pre-n* oder *Draft-n* sind bereits Geräte erhältlich, die im 2,4 GHz-Bereich arbeiten. 802.11n ist jedoch nicht auf 2,4 GHz beschränkt und soll auch im 5-GHz-Bereich eingesetzt werden.

802.11n im 2,4-GHz-Bereich ist abwärtskompatibel zu 802.11b und 802.11g, kann dann aber seine Vorteile kaum unter Beweis stellen. Die frühzeitige Investition in Geräte, die einen unfertigen Standard implementieren, ist riskant – ob sich in naher Zukunft die teure, dann vermutlich bereits veraltete Hardware auf einen aktuellen und kompatiblen Stand per Firmware- oder Treiberupdate versetzen lässt, ist unsicher. Nicht wenige Treiber oder Firmware für bereits etablierte Standards erweisen sich als unfertige Baustellen, die nie vollendet werden. Ein Test mit einer MIMO-WLAN-Karte von Sitecom (MIMO-XR WL-150 v1 001) ergab keine Verbesserung der Reichweite gegenüber einer guten WLAN-Karte nach 802.11g.

5.1.4 Bruttodatenrate und Nettodatenrate

Die für die unterschiedlichen WiFi-Standards angegebenen Übertragungsraten weichen deutlich von der tatsächlich für den Anwender zur Verfügung stehenden Bandbreite ab. Die maximale Übertragungsgeschwindigkeit be-

trägt lediglich die Hälfte der beworbenen Angaben. 802.11b erreicht maximal etwa 680 kByte pro Sekunde – das sind effektiv lediglich 5,5 MBit. WiFi mit einer Datenrate von 54 MBit erzielt bestenfalls bis zu 3,3 MByte pro Sekunde.

5.2 Geräte für den Aufbau eines Meshnetzwerks

Für die Verbindung zu einem Mesh eignen sich prinzipiell alle mit WLAN-Karten ausgestatteten Notebooks und Arbeitsplatzrechner oder dezidierte WLAN-Router. Besonders preiswerte WLAN-Karten oder USB-Dongles für Notebooks oder Desktop-Rechner sind schon für weniger als 20 Euro zu haben, falls der PC nicht wie heute üblich WLAN bereits integriert hat. Allerdings treten in einem Mesh mit vielen WLAN-Karten Probleme auf, die auf unausgereifte Treiber zurückzuführen sind. Auf diese Problematik wird im Abschnitt 5.4 ab Seite 112 näher eingegangen.

5.2.1 SoHO-Router oder PC?

Empfehlenswert für die meisten Anwendungsfälle ist die Anschaffung eines preisgünstigen WLAN-Routers aus dem SoHO-Bereich für Heimanwender, der sich mit einer alternativen Firmware zu einem Mesh-Router umrüsten lässt. Ein geeigneter WLAN-Router mit eingebautem Ethernetswitch, der sich durch eine alternative Firmware zu einem Meshknoten umrüsten lässt, kostet heute etwa 40 bis 80 Euro. Die Reichweite der WLAN-Karte in einem Router mit externer Antenne ist fast immer besser als die einer eingebauten WLAN-Karte im PC, und man kann den Router dort aufstellen, wo der Empfang des Netzwerks am besten ist – auf dem Dach, am Fenster oder auf dem Balkon. Ein wetterfestes Gehäuse für einen kleinen WLAN-Router reißt keine großen Löcher in das Budget.

Eine im Router vorhandene Firewall bietet zusätzlichen Schutz für die Computer der Netznutzer, die sich per Ethernetkabel mit dem Mesh-Router verbinden können. Der SoHO-Router benötigt nur wenig Strom und kann rund um die Uhr selbständig arbeiten, um die Netzabdeckung für andere Netzteilnehmer zu verbessern. Ein Meshnetzwerk basiert auf dem Prinzip gegenseitiger Hilfe – die Router helfen sich gegenseitig beim Transport von Daten. Wenn die Anwender ihre Nodes an exponierten Standorten aufstellen und diese permanent in Betrieb lassen, nehmen sie nicht nur als Konsumenten am Netzwerk teil, sondern geben auch wieder etwas durch die Bereitstellung von Infrastruktur zurück – das Prinzip der gegenseitigen Hilfe ist Basis eines Community-Netzwerks. Mit einem Arbeitsplatz-Computer werden das nur wenige Anwender machen wollen, da Workstations üblicherweise eine vernehmliche Geräuschkulisse haben und die Stromrechnung in die Höhe treiben.

Für die Montage an einem Mast ist ein herkömmlicher PC in einem wasserdichten Gehäuse zu sperrig – außerdem lässt dieser sich nicht ohne Weiteres mit einer ungefährlichen Kleinspannung wie ein kleiner SoHO-Router per Power-over-Ethernet betreiben. Die Montage eines herkömmlichen PCs unter dem Hausdach oder am Mastfuß erhöht die Kosten, weil dann zusätzliche Antennenkabel und Stecker angeschafft werden müssen. Diese reduzieren die Reichweite und schmälern den Geldbeutel. Außerdem muss man sich in diesem Fall mit dem Wirrwarr von instabilen Treibern und unterschiedlichen Hardwareversionen bei der Beschaffung der WLAN-Karten beschäftigen.

Natürlich hat es auch Vorteile, einen PC oder Notebook älteren Semesters als Mesh-Router zu verwenden. Außer dem guten Gefühl, dass die ausgediente Hardware noch nützlich ist, sind das vor allem die Möglichkeit, mehrere WLAN- und Ethernet-Interfaces an einem Gerät zu betreiben. Das macht sie interessant für Gateways oder zentrale Router an exponierten Stellen. Außerdem können weitere Dienste wie File- oder Webserver darauf laufen, ohne die Hardware in die Knie zu zwingen. Doch der Stress mit Stromversorgung, Konfiguration und kapriziösen WLAN-Karten wiegen diese Vorteile selten auf.

5.2.2 WLAN-Karten für PCs und Routerboards

WLAN-Karten für PCs sind heute als PCI-Karten, USB-Dongles oder USB-Adapter mit Anschlusskabel erhältlich. Für Notebooks gibt es außerdem PC-Cards nach den Standards PCMCIA Typ II (16 Bit-Schnittstelle mit 5 Volt oder 3,3 Volt Betriebsspannung), Cardbus (32 Bit 3,3 Volt) oder PC-Card Express. In modernen Notebooks findet sich praktisch immer auch ein Mini-PCI-Sockel mit eingebauter WLAN-Karte nach dem Mini-PCI-Standard.

Einige im Handel erhältliche PCI-Karten sind lediglich PCI-zu-MiniPCI-Adapter mit eingesteckter Mini-PCI-Karte unter einer Abdeckung aus Metallblech. Solche Adapter-Karten lassen sich mit einer leistungsfähigeren Mini-PCI-Karte aufrüsten. Das ist unter anderem eine preiswerte Methode, um zu einer WLAN-Karte für den WLAN-Standard 802.11a zu kommen, da Mini-PCI-Karten vergleichsweise preiswert sind.

Der Austausch ist einfach, solange die neue MiniPCI-Karte über die gleiche Antennenbuchse verfügt: Einfach den Deckel entfernen, Antennenkabel abziehen und Karten tauschen. Aber: Die Mini-PCI-Karte wird über ein kurzes Antennenkabel (Pigtail) mit dem Antennenanschluss am Slotblech der PCI-Karte verbunden. Sind die Antennenbuchsen auf der alten und der neuen Mini-PCI-Karte unterschiedlich, muss auch noch ein neues Pigtail angeschafft werden. Diese Pigtails sind relativ teuer.

PCMCIA-Karten für Notebooks mit 5 Volt und 16 Bit sind mittlerweile veraltet. Sie sind gebraucht oder als Restposten für wenig Geld zu bekommen.

Sie passen auch in alte Notebooks mit 486er Prozessoren oder besser, die sich damit ihr Gnadenbrot als stromsparende WLAN-Relais verdienen können. Gerade diese älteren Karten verfügen häufig noch über einen Antennenanschluss, um die Reichweite zu erhöhen. Beliebt sind die alten Karten mit den Chipsätzen Lucent/Orinoco oder Prism2/2.5/3 mit Antennenanschluss für 802.11b. Allerdings sind auch hier die passenden Antennenstecker oder fertig konfektionierten Adapterkabel (Pigtails) relativ teuer.

USB-Adapter sind interessant wegen der Möglichkeit, sie an einer günstigen Stelle plazieren zu können. Ein USB-Kabel darf bis zu 5 Meter lang sein, ist preiswert und verursacht keine Verluste beim Senden und Empfangen. Bei der Reichweite schneiden USB-Dongles am schlechtesten ab. Sie verwenden in der Regel stark verkleinerte Behelfsantennen, weil beim Design eine geringe Baugröße im Vordergrund steht. Wesentlich besser sind USB-Adapter mit ausklappbarer Antenne. Einige Bastler bringen USB-Adapter im Fokuspunkt einer umgerüsteten Satellitenschüssel an – damit gewinnen sie eine Antenne mit großer Reichweite und USB-Anschluss.

Fest eingebaute PCI-Karten sind für die Verbindung zum Mesh aus der Wohnung oder dem Büro ungünstig, da die Antenne unter dem Tisch an der Rückseite des Computers im elektromagnetischen Störnebel des PC sitzt. Entsprechend gering ist die Reichweite. Zusätzliche Antennen, Kabel und Stecker zur Erhöhung der Reichweite treiben die Kosten in die Höhe. Qualitativ hochwertiges Kabel und passende Stecker sind nicht gerade billig und verursachen immer zusätzliche Verluste beim Senden und Empfangen. Das Resultat einer solchen Lösung ist wahrscheinlich wesentlich schlechter als bei der Anschaffung eines geeigneten SoHO-Routers.

5.2.3 SoHO-WLAN-Router

Einige dieser Router sind sehr flexibel, denn es existieren für diese Geräte hochentwickelte alternative Linux-Betriebssysteme. Damit werden diese billigen Geräte zunehmend zur Konkurrenz von Produkten aus dem professionellen WISP-Bereich (*Wireless Internet Service Provider*).

Bei den von der Freifunk-Firmware und der OpenWRT-Version „White Russian" unterstützten Routern mit einem WLAN-Chipsatz der Firma Broadcom funktioniert der Ad-Hoc-Modus reibungslos. Die neuere OpenWRT-Version „Kamikaze" unterstützt auch viele andere Chipsätze, unter anderem Atheros SoC (*System on a Chip*) und Chipsätze von Texas Instruments. Die Broadcom-Chipsätze funktionieren erwiesenermaßen mit annehmbarer Stabilität, auch Atheros-Chips lassen sich inzwischen mit den Open-Source-Treibern des Madwifi-Projekts ansprechen und können sogar gleichzeitig als Accesspoint und als Ad-Hoc-Station arbeiten. Die Treiber für Chips von Texas Instruments sind bislang ungeeignet für den Einsatz im Mesh. Details dazu im Abschnitt 5.3 ab Seite 104.

5.2.4 Professionelle WLAN-Router

Für einige hundert Euro sind professionelle Router aus dem WISP-Bereich erhältlich. Bekannte Hersteller sind Mikrotik, Soekris Engineering und PC-Engines. Diese Firmen verkaufen Routerplatinen mit mehreren Steckplätzen für Mini-PCI-Karten und das notwendige Zubehör zum Aufbau eines Routers für den Innen- oder Außenbereich. Die Mainboards kosten etwa 100 bis 200 Euro, dazu kommen dann noch Kosten für WLAN-Karten, Antennenkabel, Antennen, Netzteil und Gehäuse. Professionelle Router sind meistens für den Betrieb über Power-over-Ethernet vorbereitet und haben mehr Rechenleistung als ein kleines SoHO-Gerät. Die Problematik der unterschiedlich gut funktionierenden Chipsätze und Treiber tritt hier jedoch genauso auf wie bei Arbeitsplatzrechnern und Notebooks.

Zu den Routerplatinen werden üblicherweise Karten mit Atheros-Chips angeboten. Kaum eine der angebotenen Lösungen hat genügend Rechenleistung, um die Übertragungsrate von zwei Atheroskarten auszulasten. Standardmäßig kommen angepasste Linuxdistributionen oder BSD-Versionen wie beispielsweise OpenBSD auf diesen Geräten zum Einsatz.

5.2.5 WLAN-Router im Selbstbau

Wenn man bereit ist, Geld für qualitativ hochwertige Antennenkabel und Stecker auszugeben oder geeignete USB-Hardware nebst Treiber zur Verfügung hat, kann man mit einem alten PC einen sehr leistungsfähigen Router aufbauen. Hier steht die enorme Vielfalt von Applikationen unter Linux oder BSD-Varianten zur Verfügung, und der Router kann mit seiner weitaus höheren Rechenleistung Daten sehr viel schneller transportieren als ein kleines SoHO-Gerät. Mehrere Netzwerkinterfaces können auch dann eingesetzt werden, wenn WLAN-Karten auf der CPU Rechenlast verursachen. Auch die Rechenlast durch ein proaktives Routingprotokoll in einem großen Meshnetzwerk ist kein Problem, und es bleiben immer noch genug Ressourcen übrig, um den Computer für andere Serverdienste zu nutzen.

Nachteilig ist natürlich die Größe der Geräte und die Stromversorgung sowie die Unzuverlässigkeit eventuell verschlissener Hardware (vor allem gealterte Elektrolytkondensatoren in Netzteilen). Am ehesten wird man ein solches Gerät auf dem Dachboden direkt unter dem Dachfirst montieren und Antennenkabel nach draußen verlegen. Als Core-Router an einem zentralen Knotenpunkt im Mesh eignet sich ein umgerüsteter PC hervorragend, insbesondere wenn über mehrere Punkt-zu-Punkt-Verbindungen ein drahtloser Backbone aufgebaut werden soll.

Kommen mehrere WLAN-Schnittstellen zum Einsatz, relativiert sich der größere Stromverbrauch und der Anschaffungspreis für qualitativ hochwertige Kabel und Stecker im Vergleich zu mehreren Embedded-Geräten recht schnell – vier SoHO-Router einschließlich ihrer Netzteile mit zusätzlichen

Antennenkabeln, Antennensteckern und leistungsfähigen Antennen sind auch nicht wesentlich sparsamer oder preiswerter als ein PC mit mehreren WLAN-Karten. Vor allem der Einsatz von USB-WLAN-Karten ist preislich attraktiv – ein USB-Kabel hat keine Verluste und kostet nicht viel.

PCs von der Leistungsklasse eines AMD K6 oder Pentium II oder III gehören für die meisten Anwender bereits zum alten Eisen, das gerne für einen guten Zweck verschenkt wird. Die gemessene Leistungsaufnahme eines solchen Rechners beträgt etwa 25 bis 30 Watt, wenn kein allzu schlechtes oder überdimensioniertes Netzteil verwendet wird.

Die Grafikkarte oder CD-Laufwerke und andere Komponenten kann man getrost ausbauen und anderweitig verwenden – sie benötigen ohnehin nur Strom, und kaum jemand wird zur Wartung einen Monitor auf den Dachboden schleppen. Für Wartungsarbeiten sollte man eine serielle Linuxkonsole einrichten und sich über ein serielles Kabel per Terminalemulation[2] mit dem Router verbinden, wenn dieser wegen einer Störung nicht mehr über seine Netzwerkinterfaces erreichbar ist. Für den Anwendungszweck als Meshrouter gibt es bereits angepasste Linuxdistributionen wie beispielsweise Pyramid-Linux[3] oder Meshlinux,[4] welche die Einrichtung eines solchen Routers vereinfachen und mehr Treiber für WLAN-Karten mitbringen als eine Linuxdistribution für Arbeitsplatzrechner.

5.2.6 HF-Steckverbinder

Seit der Einführung der ersten WiFi-Geräte sind deren Preise kontinuierlich gefallen, die für Antennen, Kabel und Stecker aber praktisch unverändert geblieben. Vor allem die Preise für geeignete WLAN-Antennenstecker und Buchsen sind überhöht. Die amerikanische Regulierungsbehörde für Kommunikationstechnologie (FCC) hat bei der Einführung der WLAN-Geräte darauf bestanden, dass dort keine genormten, handelsüblichen Hochfrequenzverbinder zum Einsatz kommen dürfen. Damit sollte der reichweitensteigernde Einsatz von gewinnbringenden Antennen oder Verstärkern unterbunden oder zumindest erschwert werden.

So kamen bald – teure – HF-Stecker mit einem weiblichen Innenkontakt und HF-Buchsen mit einem männlichen Innenkontakt in den Handel, mit der Zusatzbezeichnung „Reverse" oder „Reverse Polarity". Der Aufbau dieser HF-Verbinder (Reverse-SMA, Reverse-TNC) entspricht ansonsten den üblichen HF-Steckern und -buchsen (SMA, TNC). Ein hochwertiger Reverse-HF-Verbinder kostet etwa dreimal soviel wie ein Standardverbinder. Besonders teuer sind Miniaturstecker zum Anschluss an PC-Cards und Mini-PCI-Karten, falls sie überhaupt erhältlich sind (Hirose, MMCX, Lucent).

[2] `http://tldp.org/HOWTO/Text-Terminal-HOWTO-4.html`
[3] `http://pyramid.metrix.net`
[4] `http://open-mesh.net/meshlinux`

Reverse-Verbinder sind nur zum Anschluss an die WLAN-Geräte erforderlich, ansonsten können qualitativ hochwertige Normstecker und -buchsen verwendet werden. Verbinder nach der N-Norm sind besonders robust, verlustarm und zuverlässig. Die meisten Antennen werden über eine N-Buchse angeschlossen.

Vorsicht beim Kauf allzu günstiger Steckverbinder oder Einbaubuchsen zum Selbstbau von Antennen: WLAN arbeitet mit hohen Frequenzen und stellt entsprechend hohe Ansprüche an Kabel und Steckverbinder. Hochwertige Steckverbinder verwenden als Isolatoren Teflon, welches sehr gute Hochfrequenzeigenschaften hat und beim Löten temperaturbeständig ist. Billigramsch wird leider auch in diesem Produktbereich hergestellt und hat auch im Neuzustand nicht einmal den Schrottwert, wenn neben dem Ärger auch die verschwendete Arbeitszeit mit einkalkuliert wird. Wenn beim Löten das weiße Isolationsmaterial schon weggeschmolzen ist, bevor das Lötzinn fließt, hat man Müll eingekauft.

Auch wenn hier nicht am falschen Ende gespart wird, sind Verluste unvermeidlich. Deshalb ist es zweckmäßig, WLAN-Systeme nach Möglichkeit so zu konzipieren, dass sie ohne zusätzliche HF-Komponenten auskommen.

5.2.7 Koaxialkabel

Zur Übertragung von elektromagnetischen Wellen in WLAN-Systemen dienen abgeschirmte Koaxialkabel. Diese unterscheiden sich lediglich in ihrer Impedanz und Qualität von den abgeschirmten Koaxialkabeln, die z. B. beim TV-Empfang eingesetzt werden.

Koaxiale Leitungen bestehen aus einem Innenleiter und einer Abschirmung, zwischen denen sich eine Isolationsschicht befindet, das Dielektrikum. Die Wellenübertragung findet nicht im Metall des Innenleiters statt, sondern im Dielektrikum zwischen den Oberflächen des Innen- und Außenleiters.

Wichtige Vergleichswerte für WLAN-Kabel sind die Impedanz oder der Wellenwiderstand des Kabels in Ohm (für WLAN immer 50 Ohm) und die Verluste in Dezibel (siehe auch Abschnitt 7.1 ab Seite 170), die in der Leitung für die angegebenen Frequenzen entstehen. Da die Frequenzen von WLAN sehr hoch sind, sind auch die Ansprüche an die Leitungen hoch. Die Verluste entstehen vor allem im Dielektrikum und steigen mit der Frequenz. Als Materialien für das Dielektrikum werden vor allem Polyethylen und Luft eingesetzt. Luft ist ein verlustarmes Dielektrikum, Polyethylen hat dagegen wesentlich schlechtere Eigenschaften.

Koaxialkabel älterer Bauart haben ein massives Dielektrikum aus Polyethylen und weisen relativ große Verluste auf. Diese Kabel sind preiswert und mechanisch ziemlich robust. Ebenso wie bei Stromleitungen haben dünnere Koaxkabel wesentlich höhere Verluste als dicke.

Moderne Koaxialkabel ersetzen das massive Dielektrikum durch eines mit möglichst hohem Anteil von Luft – anstatt massivem Kunststoff werden beispielsweise Kunststoffschäume oder Luftzellen eingesetzt, die durch Kunststoffstege abgetrennt sind. Diese modernen Kabel sind sehr teuer und müssen sachgerecht behandelt werden. Auf keinen Fall dürfen sie scharf geknickt oder in irgendeiner Weise gequetscht werden. Wenn der Innenleiter an einer Stelle der Koaxialleitung nicht mehr zentrisch innerhalb des Querschnitts sitzt, kann man das Leitungsstück verschrotten. Zu den Koaxialkabeln gibt es Datenblätter – man sollte unbedingt bei der Handhabung und beim Verlegen den angegebenen minimalen Biegeradius berücksichtigen.

Die schönste Installation ist nach kurzer Zeit unbrauchbar, wenn Wasser in Koaxialleitungen und Steckverbinder eindringt und sich durch die Kapillarwirkung innen ausbreitet. Die Schwachstellen für eindringende Feuchtigkeit sind beschädigte äußere Isolationsschichten und die Übergänge zu Steckverbindern. Gecrimpte Stecker sind unbrauchbar, wenn der Aufstellungsort den Witterungseinflüssen ausgesetzt ist, da sie niemals wirklich dicht genug sind, um die Kapillarwirkung auszuschließen. Deshalb sollten im Außenbereich unbedingt solide geschraubte Stecker verwendet werden, die innen eine Dichtung haben.

Zusätzlich sollte man die Verbindungen mit selbstvulkanisierendem Spezialklebeband aus dem Elektronik-Fachhandel umwickeln. Wenn es um die zu schützenden Stellen gewickelt wird, muss es unbedingt um einen vorgegebenen Betrag gestreckt werden, der in der Gebrauchsanleitung nachzulesen ist. Diese Bandage wird zu einem enganliegenden Gummischlauch ähnlich einem Schrumpfschlauch.

Das selbstvulkanisierende Klebeband sollte mit PVC-Klebeband umwickelt werden. Manche Vögel oder andere Tiere knabbern gerne an dieser Gummimischung und können so im Laufe der Zeit den Ausfall eines Systems herbeiführen. Auf diese Weise haben Krähen in Bangladesh den Ausfall eines VSAT-Systems verursacht, durch den sehr hohe Kosten entstanden sind.

5.2.8 Gehäuse

Oft verwenden Bastler die bekannten Tupperwaredosen als wasserfeste Behälter für die Außenmontage eines SoHO-Routers. Das erweist sich besonders an heißen Sommertagen als schlechte Idee, da sich durch den transparenten Kunstoff die Luft in der Dose stark aufheizt. Ein deutliches Anzeichen dafür, dass es zu heiß hergeht, ist die sichtbare Verformung des Routergehäuses.

Selbst wenn der Router am ersten wirklich heißen Sommertag durch Überhitzung noch nicht in die ewigen Jagdgründe eingegangen ist, verschleißen die elektronischen Komponenten recht schnell. Vor allem die Elektrolytkondensatoren in der internen Spannungsstabilisierung haben nur eine

begrenzte Lebensdauer, wenn sie hohen Umgebungstemperaturen ausgesetzt sind. In den Halbleiterchips tritt außerdem bei so hohen Temperaturen verstärkt ein als *Elektromigration* bezeichneter Effekt auf: Bei hohen Temperaturen tragen die fließenden Ströme Material von den chipinternen Leiterbahnen ab, die auf diese Weise immer dünner werden. Die Chips altern unter hohen Temperaturen auf diese Weise recht schnell und erzeugen willkürliche und schwer nachvollziehbare Fehlfunktionen. Auch mit der UV-Beständigkeit einer Tupperbox ist es nicht weit her. Der Kunststoff wird wahrscheinlich nach kurzer Zeit einreißen, wenn er mechanisch belastet wird, oder einfach auseinanderfallen.

Empfehlenswert sind hingegen helle Gehäuse aus Aluminium, verzinntem Stahlblech oder UV-beständigem hellen Kunststoff, die sich unter direkter Sonneneinstrahung wenig aufheizen. Metallgehäuse sollten unbedingt mit einer hellen, deckenden Farbe lackiert sein. Auch wenn man es anders vermuten würde, heizen sich blanke oder mit Klarlack überzogenen Metallgehäuse in der Sonne stark auf. Wetterfeste weiße oder hellgraue Kunststoffgehäuse – zum Beispiel elektrische Anschlussdosen für den Außenbereich mit hoher Schutzklasse nach IP-Norm IP64 oder höher – eignen sich besser als eine Dose für das Gefrierfach. Die Tupperdose sollte man nur für hitzebeständige und einfach zu ersetzende Komponenten verwenden, z. B. für selbstgebaute Quadantennen.

Zusätzlich ist es empfehlenswert, bei Gehäusen im Außenbereich einen Sonnenschutz vorzusehen, um die Innentemperatur bei direkter Sonneneinstrahlung zu senken.

Gehäuse sollten im Außenbereich niemals hermetisch dicht sein, sondern am Boden an der tiefsten Stelle zumindest eine Entlüftungsbohrung haben. Auf wasserdichte Kabeldurchführungen kann man daher verzichten, wenn sie an der Unterseite des Gehäuses montiert sind. Versucht man ein Gehäuse möglichst dicht zu bekommen, sammelt sich darin nach längerem Betrieb unter Garantie eine größere Menge Wasser an. Ein Gehäuse – zumal eines mit mehreren Durchbrüchen für Netzwerkkabel, Stromversorgung, Antennen oder Antennenkabel – ist nie absolut dicht.

Wenn es nicht entlüftet ist, lässt sich durch Temperaturschwankungen folgender Effekt beobachten: Tagsüber erwärmt sich das Gehäuse stark, im Gehäuse entsteht durch die Ausdehnung der erwärmten Luft ein Überdruck, der irgendwie entweicht. Nachts entsteht durch die Abkühlung ein Unterdruck, kalte Luft wird nach innen gesaugt und bringt Feuchtigkeit mit. Dieser Prozess wiederholt sich, dabei entweicht tagsüber die nachts eingesaugte Feuchtigkeit nicht oder nur zu einem Teil. Durch diesen sich wiederholenden Prozess füllt sich das Gehäuse nach und nach teilweise mit Wasser, welches früher oder später die Installation durch Korrosion oder Kurzschlüsse in der Elektronik beschädigt.

5.2.9 Typische Hardware-Probleme im Ad-Hoc-Modus

Theoretisch sollte jedes Gerät, welches auf seiner Packung das Logo „WiFi-kompatibel" trägt, ohne Schwierigkeiten mit Geräten aller anderen Hersteller sowohl im Infrastruktur- als auch im Ad-Hoc-Modus zusammenarbeiten. In der Praxis funktioniert kaum eine WLAN-Karte korrekt im Ad-Hoc-Modus.

Da die meisten Anwender WLAN bislang fast ausschließlich im Infrastrukturmodus mit einem Accesspoint betreiben, vernachlässigen die Hersteller die korrekte Funktion der Ad-Hoc-Betriebsart. Dazu kommt, dass der Ad-Hoc-Modus von 802.11 einigermaßen komplex ist. Verglichen damit ist der Infrastruktur-Client-Modus weitaus einfacher zu implementieren. In Kombination mit der von vielen Firmen angewandten Bananen-Taktik (das Produkt reift beim Kunden) darf sich der Anwender bei der Verwendung der Ad-Hoc-Betriebsart in einem Meshnetzwerk auf einigen Ärger gefasst machen – und das nicht nur bei kleineren Firmen, sondern auch bei großen Herstellern wie Intel, von denen man solidere Entwicklungsarbeit erwarten würde.

Unabhängig von der verwendeten Bauform oder dem Hersteller der WLAN-Karte ist bei der Wahl der Hardware deshalb neben der Qualität vor allem der in der Karte verwendete Chipsatz kaufentscheidend. Für den Einsatz in Meshnetzwerken ist die Auswahl sehr begrenzt, da kaum ein Chipsatz auf dem Markt richtig im Ad-Hoc-Modus funktioniert. Die Probleme liegen in den Hardwaretreibern und/oder der zum Chipsatz zugehörigen Firmware. Deshalb ist es oft gleichgültig, unter welchem Betriebssystem eine WLAN-Karte eingesetzt wird.

Auch unter Open-Source-Betriebssystemen liegt die Firmware nur als Binärdatei vor und ist urheberrechtlich geschützt. Wenn sie manipuliert werden könnte, wäre bei vielen WLAN-Karten der Betrieb auf nicht zugelassenen Frequenzen oder mit überhöhter Sendeleistung möglich. Daher werden die Quelltexte und Details der Firmware von der Herstellern unter Verschluss gehalten und können nicht legal verändert werden. Man ist also bei Firmwareproblemen von der Bereitschaft der Hersteller abhängig, Fehler zu beheben.

Bis heute arbeitet beispielsweise die Firmware für die WLAN-Karten von Intel mit dem Chipsatz IPW2200 im Ad-Hoc-Modus fehlerhaft. Da dieser Chipsatz mittlerweile schon wieder Schnee von gestern ist, steht zu befürchten, dass eine funktionierende Firmware nie verfügbar sein wird.

Es ist bei der Mehrheit der PC-Notebooks reine Glücksache, ob sich eine brauchbare Verbindung zu einem Meshnetzwerk herstellen lässt. Die Anwender vermuten häufig schlechten Empfang als Ursache bei nicht funktionierenden Verbindungen – dabei sind allzu oft Fehler in den Treibern oder in der Firmware verantwortlich. Deutlich wird das, wenn mehrere Ge-

räte auf dem gleichen Tisch stehen und partout nicht miteinander „reden" wollen. Welcher Besitzer eines Autos würde es akzeptieren, wenn die Werkstatt den defekten Rückwärtsgang in einem Neuwagen mit einem Schulterzucken quittiert und den Mangel als Selbstverständlichkeit betrachtet, mit der man sich abzufinden hat? „Na, bitte! Vorwärts fährt er doch!"

Ein möglicher Weg zur Abhilfe bestünde darin, dass mehr und mehr Kunden konsequent auf ihre Rechte als Verbraucher bestehen. Schließlich ist ein Auto, das einen defekten Rückwärtsgang hat, ebenso wie eine WLAN-Karte, die nicht im Ad-Hoc-Modus arbeitet, ein defektes Produkt, auf dessen Reparatur man bestehen kann. Ist der Hersteller nicht in der Lage, ein funktionierendes Produkt anzubieten, ist man selbstverständlich berechtigt, vom Kaufvertrag zurückzutreten.

Die Situation bei Linux-Treibern

In der Linuxwelt sieht die Treibersituation im Augenblick eher schlecht au auss. In der Vergangenheit haben die Entwickler von WLAN-Treibern die Funktionalität von 802.11 für die meisten Chipsätze auf eigene Faust implementiert und damit das Rad immer wieder neu erfunden. Vielfach mangelt es den Entwicklern dabei am grundsätzlichen Verständnis von IEEE 802.11 – der Standard umfasst einige hundert Seiten und genügend Diagramme, um eine Wohnung zu tapezieren.

Inzwischen kann man auf eine Verbesserung der Situation hoffen, da es einen neuen Ansatz im Linuxkernel gibt. Statt jedem Programmierer die Implementierung von 802.11 selbst zu überlassen, gibt es mittlerweile einen soliden 802.11-Stack (d80211, inzwischen umbenannt in mac80211), den die Firma Devicescape unter GPL gestellt hat und der ab Kernelversion 2.6.22 oder 2.6.23 Bestandteil des Linuxkernel werden soll. Inzwischen arbeiten einige WLAN-Treiber-Projekte daran, ihre Software für den neuen 802.11-Stack anzupassen.

Unter Linux muss ein Treiber nicht für eine spezifische Karte eines bestimmten Herstellers installiert werden, wie man es von der Windows-Welt gewohnt ist. Es ist also nicht so, dass ein Treiber für die Karte der Marke XY sich weigert, mit der Karte der Marke WZ zu funktionieren, obwohl beide vom gleichen OEM-Hersteller stammen. Entscheidend ist, ob es für den in der Karte verwendeten Chipsatz einen funktionierenden Treiber gibt und ob Linux den von der Karte ausgegebenen Typcode einem bestimmten Chipsatz zuordnen kann.

Einige Chiphersteller wie Zydas, Realtek und Ralink bieten selbst Linux-Treiber als Sourcecode an, doch dieser kompiliert oft nicht gegen den aktuellen Linuxkernel. Da jener sich fortlaufend weiterentwickelt, müssten auch die Treibersourcen an neuere Versionen des Kernels angepasst und gewartet werden. Eine Arbeit, die die Hersteller meist nicht tun und für die

sich manchmal auch kein freier Entwickler findet. Doch immerhin stoßen die Hersteller damit aktiv die Entwicklung von Treibern für Linux an. Gute Hardware hat auch einige Zeit, nachdem sie vom Markt verschwunden ist, noch ihre Fans – so kommen die Anwender oft zu einer Stabilität und Funktionsvielfalt, die proprietäre Treiber nur bei wenigen, wirklich seriösen Firmen erreichen.

Ndiswrapper – Windows-Treiber unter Linux als Behelfslösung

Sofern der Hersteller einen gut funktionierenden Treiber mit brauchbarer Firmware für Windows bereitstellt, kann bei fehlendem Linux-Treiber als Notlösung das Programm Ndiswrapper weiterhelfen. NDIS (*Network Device Interface Specification*) ist eine von Windows vorgegebene Schnittstellenspezifikation für Netzwerkkarten. Ndiswrapper (*to wrap* bedeutet auf Englisch *einwickeln*) arbeitet als Kompatibilitätsschicht, die als Vermittler zwischen dem Linuxkernel und dem Windowstreiber fungiert. Damit lassen sich die Funktionen der Netzwerkkarte durch den Windowstreiber unter Linux nutzen. Dazu benötigt man den Treiber für Windows XP, die Firmware (falls bei der Initialisierung der Karte ein Firmwareupload in die Karte erforderlich ist) und die dazugehörende Installationsdatei mit der Endung `.INF`. Ndiswrapper besteht aus dem Dienstprogramm `ndiswrapper` und einem gleichnamigen Kernelmodul.

Auf der Webseite des Ndiswrapper-Projekts ist die Hardware aufgelistet, die mit Ndiswrapper erfolgreich getestet wurde. Auch die passenden Treiber- und Firmwaredateien sind dort zu finden. Sollte die betreffende Karte dort noch nicht aufgelistet sein, lohnt sich ein Versuch mit den Windowstreibern von der CD des Anbieters. Die erforderlichen Dateien sind normalerweise in einem Cab-Archiv oder einem selbstextrahierenden Zip-Archiv verpackt. Cab-Dateien lassen sich mit dem Befehl `cabextract` unter Linux auspacken, selbstextrahierende Zip-Archive (Dateiendung `.exe`) können mit dem Befehl `unzip` geöffnet werden. Gelingt das nicht, führt man die Installation auf einem Windowssystem durch. Anschließend verrät der Windows-Gerätemanager, welchen Treiber die Hardware verwendet.

Die benötigten Dateien aus dem Windows-System kopiert man unter Linux zunächst in ein beliebiges Verzeichnis. Den Befehl

```
ndiswrapper -i Inf-Datei-des-Windowstreibers.inf
```

führt man mit Administratorrechten in dem Verzeichnis aus, das die Treiberdateien enthält. Er installiert den Windowstreiber in Ndiswrapper. Das Kommando

```
ndiswrapper -l
```

listet alle in Ndiswrapper installierten Treiber auf und zeigt an, ob das betreffende Gerät existiert. Nach dem Laden des Kernelmoduls `ndiswrapper.ko` mit dem Befehl

```
modprobe ndiswrapper
```

ist die WLAN-Karte in der Regel auf Anhieb betriebsbereit. Die Lösung, Windowstreiber unter Linux einzusetzen, hinterlässt bei manchen Linuxfans ein zwiespältiges Gefühl. Zwar ist es schön, das betreffende Gerät damit betreiben zu können, doch eigentlich wünscht man sich einen „richtigen", quelloffenen Linux-Treiber. Ndiswrapper funktioniert außerdem nur auf den Hardwareplattformen, für die es einen Windowstreiber gibt.

5.3 Chipsätze

Wenn der folgende Abschnitt auf die Eigentümlichkeiten und die Treibersituation der verschiedenen Chipsatz-Familien eingeht, liegt der Schwerpunkt dabei auf Linux, da Mesh-Routing eine Domäne von Linux ist. Nur wenige Mesh-Protokolle laufen auf anderen Betriebssystemen.

Prinzipiell kann sich die hier beschriebene Situation der Treiber oder der Firmware jederzeit ändern. Praktisch hat sich in den letzten Jahren nur sehr wenig getan. Bei Chipsätzen, die kaum noch im Handel sind, kann man auf eine Besserung am ehesten dann hoffen, wenn die Probleme nicht an der Firmware liegen und sich engagierte Open-Source-Entwickler der Sache annehmen.

5.3.1 Prism für 802.11b

Die Prism-Chipsätze 2, 2.5 und 3 der Firma Intersil für 802.11b waren vor wenigen Jahren die beliebtesten auf dem Markt. Vor allem von der taiwanesischen Firma Senao gab es lange Zeit exzellente Mini-PCI- und PCMCIA-Karten mit sehr hoher Reichweite. Die Senao-Karten gehörten auch zur Standardausstattung des Mesh-Routers Mesh-Cube (später umbenannt in Accesscube) der Firma 4G-Systems. Bekannt wurden die Prism-Karten wegen des Host-AP-Treibers für Linux. Prism-Karten arbeiten mit diesem Treiber im Accesspoint-, Station-, Bridge-, Ad-Hoc-, Ad-Hoc-Demo- und Monitor-Mode. Die Entwickler des Host-AP-Treibers lieferten auch gut funktionierende Utilities zum Flashen der Firmware (Prism-Utils).

Leider ist die Firmware in diesen Karten nicht robust genug gegen Cell-Splitting (siehe Seite 112). Gerade jüngere Versionen der Firmware haben damit Probleme. In WLAN-Karten im Compact-Flash-Format mit 11 MBit findet man diesen Chipsatz noch gelegentlich im Handel. Prism2, 2.5, 3 ist

bis heute eine ausgezeichnete Wahl für lange Funkstrecken im 2,4-GHz-Band.

Der Linuxkernel unterstützt diese Karten, allerdings nicht die USB-Variante dieses Chipsatzes. USB-Karten mit Prism2–3 arbeiten mit dem WLAN-NG-Treiber. Dieser wird schon seit Jahren nicht mehr weiterentwickelt und arbeitet nicht mit den Wireless-Tools unter Linux zusammen. Zudem ist die Bedienung von WLAN-NG umständlich.

5.3.2 Lucent-Orinoco

Der Orinoco-Chipsatz für 802.11b ist sehr alt und mittlerweile Geschichte. WLAN-Karten mit diesem Chipset (es existieren zwei PCMCIA-Gehäusevarianten) erkennt man praktischerweise fast immer an der Gehäuseform (Abbildung 5.1). Sie verfügen über einen Antennenanschluss gepaart mit einer hervorragenden Empfindlichkeit.

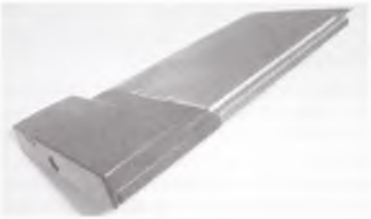


Abbildung 5.1: Typische Gehäuseform der Orinoco-Karte

Obwohl die Sendeleistung typischerweise nur 35 Milliwatt beträgt, ist die Reichweite enorm – der Hersteller gibt 600 Meter bei 1 MBit im Freien an. Einige alte Weltrekorde für Langstreckenverbindungen wurden mit diesem Kartentyp erzielt, und zwar ohne das Sendesignal mit einem Verstärker anzuheben. Die Treiberunterstützung für praktisch jedes Betriebssystem ist exzellent, sogar für ein steinaltes WindowsCE 2.0 findet man im Internet einen Treiber.

Die Orinoco-Karten waren die ersten WLAN-Karten, für die es einen sehr guten Linux-Treiber gab. Dieser stammt von Jean Tourrilhes, der auch die Wireless-Tools für Linux entwickelt hat. Die Karte beherrscht Ad-Hoc-, Ad-Hoc-Demo-, Monitor-, Infrastruktur-Client- und mit angepasster Firmware auch den Accesspoint-Modus. In vielen alten Accesspoints findet man einen PCMCIA-Slot mit einer eingesteckten Orinoco-Karte, unter anderem in der alten Apple-AirPort-Basestation für 802.11b.

Diese Karten wurden in großen Stückzahlen hergestellt und von vielen namhaften Hardwareherstellern unter eigenem Label verkauft. Gebraucht sind

sie bei Ebay zu bekommen. Karten, die als Orinoco-Karten angeboten werden, sind leider meistens stark überteuert – 50 Euro plus Versandkosten sind einfach utopisch. Wer sich Zeit nimmt, findet für wenig Geld eine der zahllosen Varianten unter einem anderen Namen.

Für diese Karten gibt es unterschiedliche Firmware-Varianten, die unterschiedlich gut im Ad-Hoc-Modus arbeiten. Einige Firmware-Versionen sind recht zickig und neigen zu Cell-Splitting. Man kann die Firmware mit einem Windows-Programm flashen, viele Originaltreiber für Windows bringen ein Flashprogramm mit. Leider hält sich die Firmware an den Standard, und man kann man die Cell-ID im Ad-Hoc-Modus nicht setzen.

5.3.3 Prism54 Hard- und Soft-MAC

Die Performance der Hard-MAC-Variante der als Prism GT und Prism Indigo (ISL3880 und ISL3890) bezeichneten Chipsätze in einem Mesh ist ausgezeichnet. Leider kommt auch bei diesen Karten gelegentlich Cell-Splitting vor. Treiber für WLAN-Karten, die auf diesem Chipsatz aufbauen, befinden sich ebenfalls im Linuxkernel. USB-Geräte mit Prism54 werden nicht unterstützt, es gibt zwar einen experimentellen Treiber, aber dieser arbeitet bestenfalls im Infrastrukturmodus.

Seit dem Verkauf von Intersil an Connexant sind Karten mit dem Prism54-Chipsatz nur noch in der billigen Soft-MAC-Variante im Handel zu bekommen. Die MAC-Schicht (die Abkürzung steht für *Media Access Control*) wird bei diesen Chipsätzen nicht in der Hardware, sondern in Software implementiert, die von der CPU des Systems, in dem die Karte installiert ist, ausgeführt wird. Die Soft-MAC-Varianten des Intersil-Chipsatzes (ISL388x, ISL389x) funktionieren unter Linux überhaupt nicht im Ad-Hoc-Modus.

Bei den Prismkarten mit Hard-MAC kann die Cell-ID vor dem Aktivieren des Interfaces (im Beispiel auf 02:03:04:05:06:07) gesetzt werden:

```
iwconfig wlan0 ap 02:03:04:05:06:07 essid olsr.freifunk.net
```

5.3.4 Atheros

Die Atheros-Chipsätze AR50XX für 802.11a, b und g arbeiten unter Linux mit dem Treiber des Madwifi-Projekts. Der Madwifi-Treiber ist eine echte Wundertüte mit vielen faszinierenden Möglichkeiten. Es gibt nichts, was – theoretisch – nicht geht: Ad-Hoc- und Ad-Hoc-Demo-Modus, Bridging, Monitoring, die Konfiguration als Accesspoint oder als Accesspoint-Client. Es ist beispielsweise möglich, eine Atheros-Karte gleichzeitig als Accesspoint und als Accesspoint-Client auf dem gleichen Kanal zu betreiben oder auch als Accesspoint und gleichzeitig als Ad-Hoc-Station. So kann man mit nur

einer WLAN-Karte per Mesh die Verbindung zum Internet herstellen und gleichzeitig einen Hotspot betreiben, der lokale Clients im Infrastrukturmodus versorgt. Diese Funktion ist bislang experimentell, und man sollte sich darauf einstellen, dass ihre Anwendung den einen oder anderen Absturz – mitunter auch des kompletten Computers – verursacht.

Wissenschaftler am MIT und an der Technischen Universität Berlin verwenden für die Forschungsprojekte MIT-Roofnet und Berlin-Roofnet einen eigenen modifizierten 802.11-Stack für das Meshing. Für Tüftler und Bastler ist Madwifi also eine tolle Spielwiese. Leider sind die Bugs ähnlich zahlreich wie die Features, was angesichts der Komplexität des Treibers wenig verwundert. Hinter dem Treiberprojekt steht jedoch eine aktive Entwicklergemeinde, und die Anpassung des Treibers für den 802.11-Stack von Devicescape ist in Arbeit.

Für Linux-Neulinge ist der Weg zu einem einigermaßen gut funktionierenden Atheros-Treiber leider steinig. Bis heute muss man den Madwifi-Treiber[5] selbst kompilieren und vorher noch mit einem Patch von den Freifunkern versehen, damit der Ad-Hoc-Modus einigermaßen funktioniert.[6] Der gepatchte Madwifi-Treiber läuft stabil, aber noch nicht optimal, und es wird fleißig am Ad-Hoc-Modus gebastelt. Deshalb sind die Atheros-Karten eine Empfehlung wert, zumindest für PC-Hardware. Mit dem Kommando

```
iwconfig wlan0 ap 02:CA:FF:EE:BA:BE
```

lässt sich die Cell-ID einfach festlegen, ohne das Interface vorher zu deaktivieren.

Aktuelle Notebooks von Apple sind mit Atheros-Karten ausgestattet. Unter Mac-OSX funktioniert der Ad-Hoc-Modus auch in einem Mesh stabil.

Ein Test des Atheros-Treibers für Windows 2000 und XP endete damit, dass die Windows-Registry von Hand editiert werden musste, um den instabilen Treiber, der immer wieder Systemabstürze verursachte, zu entfernen. Im Ad-Hoc-Modus gibt es unregelmäßig den berüchtigten blauen Bildschirm, und die Karte lässt sich nicht mehr sauber automatisch deinstallieren.

Atheros-Chipsätze führen viele Aktionen auf der Host-CPU des Rechners aus und erzeugen somit eine recht hohe CPU- und Interrupt-Last. SoHO-Accesspoints mit Atheros-Chipsätzen sind schon mit einer einzigen WLAN-Karte vollkommen überlastet, wenn sie auch nur einen Teil der angegebenen Bitrate verarbeiten müssen.

So ergab ein Test mit einem Netgear-WGT634U-Accesspoint, dass selbst der Download eines Datenstroms von Nullen, den der Linuxkernel direkt auf dem Accesspoint über das virtuelle Gerät `/dev/zero` erzeugte, maximal 1,2 MByte pro Sekunde erreichte – im Abstand von rund einem Meter

[5] `http://madwifi.org`

[6] Bereits für Ad-Hoc gepatchte Sourcen des Madwifi-Treibers und Patches gibt es unter `http://open-mesh/misc/madwifi`.

zum Accesspoint. Eigentlich sollten es mehr als drei MByte sein; jedenfalls erreicht die Karte diesen Durchsatz, wenn sie in einem PC steckt. Ein Meshrouter auf der Basis eines PC ist also für eine Atheros-Karte ein geeigneterer Einsatzort als ein Embedded-System.

Abbildung 5.2: Atheros-OEM-Karten mit Cardbus-Interface: hinten die alte, davor die neue Gehäuseform

Auch professionelle Router mit schnellerer CPU haben mit diesen Karten arge Probleme, zumal dort meistens zwei oder mehr im System stecken. Praktisch erreichen auch sie kaum die angegebene Performance. Auf der Habenseite stehen die große Funktionsvielfalt des Madwifi-Treibers und eine sehr gute Empfangsleistung. Mini-PCI-Karten mit diesem Chipsatz gibt es mit Sendeleistungen bis zu 0,4 Watt für 802.11a – damit haben diese Karten eine extrem große Reichweite für Langstreckenverbindungen. Mit den Utilities des Madwifi-Treibers lässt sich das Timing von 802.11 so justieren, dass auch Verbindungen über sehr große Entfernungen möglich sind (Hintergründe zum Timing in Abschnitt 7.6.5 ab Seite 187).

5.3.5 Atmel

Atmel-Chips für WLAN mit der Bezeichnung at76cXX für 802.11b sind nicht mehr im Handel. Das ist schade, da die Treibersituation sehr gut ist. Die Unterstützung für USB-WLAN-Karten ist für den Linuxkernel 2.4 ausgezeichnet und funktioniert im Ad-Hoc-Modus stabil. Die USB-Treiber werden nicht mehr gewartet, deshalb wird es mit neueren Kernelversionen immer schwieriger, USB-Geräte von Atmel zum Laufen zu bekommen. PCMCIA- und PCI-Karten werden dagegen auch vom aktuellen Linuxkernel 2.6 unterstützt. Auch diese Treiber funktionieren in einem Mesh gut. Die Empfangsleistung dieser Karten ist allerdings nicht überragend.

5.3.6 Broadcom

Der Chiphersteller Broadcom stellt eine funktionierende Firmware und einen Linux-Treiber bereit, der erwiesenermaßen reibungslos im Ad-Hoc-Modus arbeitet. Chipsätze dieses Herstellers werden in den meisten SoHO-

WLAN-Routern mit Linux verwendet, der Treiber ist jedoch Closed Source und steht paradoxerweise nicht für Linux-PCs und -Notebooks mit Intel-kompatiblen Prozessoren zur Verfügung.

SoHO-Router mit Broadcom-Chipsatz lassen sich mit der Freifunk-Firmware und der OpenWRT-Version „White Russian" betreiben und arbeiten im praktischen Betrieb in einem Mesh selbst unter wirklich rauhen Bedingungen sehr zuverlässig. Ein Open-Source-Projekt arbeitet seit einiger Zeit an einem freien Broadcom-Treiber für Linux. Der Open-Source-Treiber verwendet den neuen 802.11-Stack von Devicescape im Linuxkernel.

Durch einen Hack der OpenWRT-Entwickler ist es auch bei diesen Chips möglich, die Cell-ID einer Ad-Hoc-Zelle festzulegen. Auch das Timing von 802.11 kann justiert werden, so dass Langstreckenverbindungen möglich sind. Der Weltrekord über 381 Kilometer wurde auf diese Weise aufgestellt.

Broadcom-Karten befinden sich auch in älteren Computermodellen von Apple. Die Treiber sind für Mesh-Verbindungen brauchbar.

5.3.7 Cisco

Die für ihre professionelle Netzwerkhardware bekannte Firma Cisco stellte eigene Chipsets für 802.11b mit der Bezeichnung 34X/35X/4500/4800 her, die nur in wenigen WLAN-Karten auf dem Markt verwendet wurden. Cisco hat diese Karten unter dem Namen Aironet verkauft. Außerdem befinden sich Cisco-Chips in Netzwerkkarten, die als OEM-Produkte unter anderem von Xircom und Dell gelabelt wurden.

Die Treiber im Linuxkernel sind im Infrastrukturmodus über jeden Zweifel erhaben. Im Ad-Hoc-Modus funktioniert die Karte scheinbar gut, allerdings stört sie das Netzwerk. Es lässt sich beobachten, dass andere WLAN-Karten ihre Verbindung zur Ad-Hoc-Zelle verlieren und es zu Cell-Splitting kommt, sobald eine Cisco-Karte aktiviert wird.

5.3.8 Intel

Der alte Intel-Chipsatz für 802.11b (ipw2100) arbeitet in einem Mesh zuverlässig. Das gilt leider nicht für den Chipsatz ipw2200 für 802.11bg. Die Karte verliert früher oder später den Kontakt zur Funkzelle und ist nicht mehr in der Lage, ihn wiederherzustellen. Gleichzeitig schreibt der IPW-Treiber unter Linux immer wieder die Meldung `Firmware error, restarting` in `/var/log/syslog`. Die Ursache der Probleme liegt eindeutig in der Kartenfirmware. Ist der erste Firmware-Fehler aufgetreten, hilft nur noch ein Kaltstart der Karte. Bei einer Mini-PCI-Karte bedeutet das: den Rechner herunterfahren, ausschalten und neu starten, gepaart mit der Hoffnung, dass es nach dem nächsten Neustart besser klappt.

Unter Windows sind die Symptome die gleichen – die Intel-Karte verbindet sich mit einer bestehenden Funkzelle nur dann, wenn der Mond im dritten Haus des Aszendenten Käfer steht. Dieses Verhalten tritt vor allem bei schlechtem Empfang auf.

Bei dem neuen Chipsatz ipw3945 für 802.11a, b und g ist die Situation ähnlich. Der ältere Treiber ipw3945 funktioniert nicht so recht, häufig tritt Cell-Splitting auf. An einem neuen Treiber mit dem 802.11-Stack von Devicescape wird gearbeitet, dieser war bei Erscheinen dieses Buches aber noch nicht brauchbar. Karten mit diesem Chipsatz gibt es nur nach dem neuen *PCI-Express-Mini-Card*-Standard.

Intel unterstützt aktiv die Entwicklung von Linux-Treibern. Es wäre schön, wenn Treiber und Firmware eines Tages auch in einem Mesh im Ad-Hoc-Modus funktionieren würden.

5.3.9 Ralink

Die Chipsätze von Ralink für 802.11b (rt2400) und 802.11bg (rt2500) funktionieren unter Linux relativ gut. Der Hersteller selbst bietet Linux-Treiber als Sourcecode für alle Chipsätze an, auch für 802.11 Draft-n, den die Chipsätze rt2600 und rt2800 unterstützen.

Alternative Treiber gibt es vom Open-Source-Projekt rt2x00. Es arbeitet zur Zeit an zwei Treiber-Generationen für die Ralink-Chips. Zum einen kümmern sich die Programmierer um die Weiterentwicklung und Fehlerbereinigung der von Ralink veröffentlichten Sourcen (Legacy-Generation). Auf der anderen Seite sind sie dabei, eine neue Treibergeneration für den Devicescape-Stack zu schreiben, die Bestandteil des Linuxkernels werden wird.

Die von Ralink veröffentlichten Treiberquelltexte kompilieren nicht mit aktuellen Versionen des Linuxkernels. Die Treiber der Legacy-Generation lassen sich dagegen problemlos kompilieren und funktionieren bei Infrastruktur-Clients. Wie immer erscheint den Programmierern der Ad-Hoc-Modus weniger wichtig. Hoffentlich erreichen die Treiber am Ende auch dafür eine brauchbare Reife. Am ehesten dürfte dieser Zustand erreicht werden, wenn die Portierung für mac80211 abgeschlossen ist.

In preiswerten USB-Dongles wird aktuell recht häufig der Chipsatz rt2570 für USB-Dongles (erhältlich unter anderem von Hama für rund 20 Euro) verbaut. Im praktischen Test mit dem alten Open-Source-Treiber und einem Hama-USB-Dongle mit rt2570 traten noch Schwierigkeiten bei der Auswahl der Übertragungsrate im Ad-Hoc-Modus auf.

Eine Cardbus-Karte mit dem Ralink-Chipsatz rt2600 (Sitecom Typ WL-150 Version 1) mit MIMO lässt sich unter Linux bereits mit dem Legacy-Treiber des rt2x00-Projekts betreiben. Der Ad-Hoc-Modus funktionierte bei ersten Tests bereits ein bisschen (die Karte assoziierte sich, auch Ping funktionier-

te). Unter Windows 2000 arbeitet die Karte im Ad-Hoc-Modus ohne Probleme.

5.3.10 Realtek

Der letzte von Realtek als Open-Source angebotene Linux-Treiber für den Chipsatz RTL8180L (802.11b) stammt vom April 2005. Das Kompilieren des Treibers gegen einen aktuellen 2.6er Kernel schlägt wegen der betagten Softwareversion fehl, danach sind auch gleich die entpackten Quelltexte gelöscht.

Auch für diesen Chipsatz gibt es einen alternativen Open-Source-Treiber. Er ist so alt wie die letzte Veröffentlichung von Realtek. In der Download-Beschreibung steht, dass der Ad-Hoc-Modus „grob" funktioniert. Da der Chipsatz veraltet ist, wird sich daran wohl kaum noch etwas ändern.

Auch die Sourcen zu den Chipsätzen für 802.11a, b und g, RTL8185L und RTL8187L, ließen sich auf dem System der Autorin nicht kompilieren. Mit keinem Realtek-Chipsatz lässt sich unter Linux wirklich etwas anfangen.

5.3.11 Texas Instruments

Ebenso wie Broadcom verfügt Texas Instruments über WLAN-Treiber für Linux, die den Herstellern von WLAN-Routern nur als Closed-Source zur Verfügung stehen. Gleichzeitig werden WLAN-Karten mit Texas-Instruments-Chipsätzen für Arbeitsplatzrechner verkauft, für die der Hersteller keine Linux-Treiber anbietet. Texas Instruments hat auch nichts unternommen, um die Entwicklung eines Open-Source-Treibers in irgendeiner Art und Weise zu unterstützen. Trotzdem fand sich eine Gruppe von Entwicklern, die durch mühsames Reverse-Engineering versucht hat, einen benutzbaren Treiber zu entwickeln.

Allerdings funktioniert auch der Closed-Source-Treiber nicht im Ad-Hoc-Modus, andernfalls würde es schon lange viele weitere Freifunk-Router auf Basis der AVM-Fritz-Box geben.

Beim Laden des GPL-Treibers lassen ihn seine Entwickler die Meldung ausgeben, dass man sich doch besser eine Karte mit dem Chipsatz eines anderen Herstellers kaufen sollte. Die Anschaffung einer Karte mit Texas-Instruments-Chips für Linux oder Mesh-Routing ist daher reine Geld- und Zeitverschwendung.

5.3.12 Zydas

Auch die Firma Zydas hat Open-Source-Treiber für ihre Chipsätze ZD1201 für 802.11b und ZD1211/ZD1211B für 802.11a, b und g veröffentlicht. Der Treiber für das Chipset ZD1201 ist Bestandteil des 2.6er Linuxkernels. Für

die Chipsets ZD1211/ZD1211B gibt es drei Treiber unter der GPL: Der Originaltreiber des Herstellers wird seit der Übernahme von Zydas durch Atheros nicht mehr weiterentwickelt und steht auch nicht mehr zum Download bereit. Weiterhin existiert eine von Open-Source-Programmierern verbesserte Version des Originaltreibers, dessen Entwicklung ebenfalls zum Stillstand gekommen ist, und ein dritter, von Grund auf neu geschriebener GPL-Treiber mit der Bezeichnung zd1211rw.

Die Situation ähnelt damit der bei Ralink. Der neue Treiber zd1211rw ist seit der Kernelversion 2.6.18 Bestandteil der 2.6er Kernel-Sourcen und verwendet den WLAN-Stack mac80211 von Devicescape. Den letztgenannten Treiber bezeichnen die Entwickler als stabil, auch wenn es derzeit noch signifikante funktionelle Einschränkungen gibt.

5.4 Typische Hardware-Probleme im Ad-Hoc-Modus

5.4.1 Cell-Splitting

Der Dauerbrenner unter den Problemen im Ad-Hoc-Modus ist das sogenannte Cell- oder IBSS-Splitting. Zur Kennung einer Funkzelle gehört eine Identifizierungsnummer (ID), diese wird bei der Übertragung von Bakensignalen oder Beacons mitgesendet. Anhand der Bakensignale sind die Karten in der Lage, vorhandene Netze zu erkennen. In Infrastrukturnetzen wird die Identifizierungsnummer der Funkzelle als *Basic Service Set Identifier* (BSSID) bezeichnet und entspricht der MAC-Adresse der WLAN-Karte des Accesspoints. Auf Grund der klaren Hierarchie in einer Infrastrukturzelle ist das Verfahren zur Bestimmung der Zellennummer einfach: Sie wird vom Accesspoint vorgegeben.

In Ad-Hoc-Netzen heißt die Identifizierungsnummer der Funkzelle IBSSID (*Independent Basic Service Set Identifier*). Da es in einer Ad-Hoc-Zelle keine Hierarchie gibt, müssen die Ad-Hoc-Stationen selbst aushandeln, wer die Zellennummer bestimmt. Das Verfahren zur Bestimmung der Cell-ID ist wegen der fehlenden Hierarchie etwas komplizierter. Protokoll 802.11 sieht vor, das Problem durch das Versenden von Zeitstempeln zu lösen.

Aktiviert man eine WLAN-Karte im Ad-Hoc-Modus, sollte sie sich umhören, ob bereits andere Karten mit demselben Netzwerknamen (der ESSID - *Extended Service Set ID*) auf dem vorgesehenen Kanal eine Funkzelle gebildet haben und Bakensignale senden. Lässt sich noch keine Funkzelle mit einer bestehenden ISBSSID ausmachen - und nur dann - sollte die Karte eine neue Funkzelle gründen, nachdem sie sich auch auf allen anderen verfügbaren Kanälen erfolglos nach Bakensignalen für die angegebene ESSID umgehört hat.

Eine Ad-Hoc-Station, die alleine funkt, weil sie keinen Funkkontakt zu anderen hat oder als erster Knoten die Funkzelle gründen muss, erzeugt per Zufall eine IBSSID und beginnt, einen Zeitstempel hochzuzählen, der mit den Beacons übertragen wird. Später hinzukommende Stationen haben die ältere, vorgegebene ID zu verwenden und deren Zeitstempel zu übernehmen, da ihr eigener Zeitstempel jünger ist. Dieser Prozess heißt *IBSS Merge*. Empfangen andere Stationen die geänderte IBSSID mit den älteren Zeitstempel in den Beacons, stellen sie sich ebenfalls um. Damit ist sichergestellt, dass sich einzelne Karten oder Funkzellen zu einer großen, gemeinsamen Funkzelle verbinden können.

Dieses Verfahren ist schlüssig und einfach. Es funktioniert auch – solange die WLAN-Karten tun, was sie sollen. Genau an dieser Stelle kracht es im praktischen Betrieb eines Ad-Hoc-Netzes aber ständig, falls die Betreiber des Meshnetzwerks nicht in die Trickkiste greifen. Die leidvolle Erfahrung ist, dass fünf Notebooks mit fünf verschiedenen WLAN-Karten, die auf dem gleichen Tisch stehen, fünf verschiedene Ad-Hoc-Zellen gründen. In der Anfangszeit der Freifunk-Meshnetze war dies neben untauglichen Routingprotokollen das Hauptproblem.

Ad-Hoc-Stationen mit unterschiedlichen IBSSIDs ignorieren sich gegenseitig, wenn sie sich nicht auf eine gemeinsame IBSSID einigen können. Verwenden die Karten unterschiedliche IBSSIDs, dann funken einzelne oder Gruppen von Stationen isoliert nebeneinander her. Sie verwenden zwar den gleichen Kanal und dieselbe ESSID, haben aber unterschiedliche Cell-IDs und sind somit logisch getrennte Funkzellen.

Unter Linux zeigt der Befehl `iwconfig` ohne die Übergabe zusätzlicher Optionen den WLAN-Status aller Netzwerkkarten im System an. Klappt die Verbindung nicht und ist die Cell-ID unterschiedlich, obwohl ESSID, Kanal, IEEE-Standard und WLAN-Mode stimmen, liegt Cell-Splitting vor:

```
rechner_A# iwconfig
eth0    IEEE 802.11b  ESSID:"olsr.freifunk.net"  Nickname:""
        Mode:Ad-Hoc  Frequency:2.457 GHz  Cell: 02:CA:FF:EE:BA:BE
        Bit Rate:11 Mb/s   Tx-Power=15 dBm
        Retry limit:8   RTS thr=256 B   Fragment thr=512 B
        Encryption key:off
        Power Management:off
        Link Quality:12/100  Signal level:14/100
        Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
        Tx excessive retries:0  Invalid misc:0   Missed beacon:0

rechner_B# iwconfig
eth0    IEEE 802.11b  ESSID:"olsr.freifunk.net"  Nickname:""
        Mode:Ad-Hoc  Frequency:2.457 GHz  Cell: 02:0B:6B:20:22:FE
        Bit Rate:11 Mb/s   Tx-Power:16 dBm   Sensitivity=0/3
        Retry limit:8   RTS thr=256 B   Fragment thr=512 B
        Encryption key:off
```

```
Power Management:off
Link Quality:12/100  Signal level:14/100
Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
Tx excessive retries:0  Invalid misc:0   Missed beacon:0
```

Im günstigsten Fall kann sich eine nicht richtig funktionierende Karte nicht mit der vorhandenen Funkzelle verbinden und sendet allein und isoliert auf ihrer privaten Funkinsel Bakensignale aus. Die Anwender wundern sich dann, warum der Zugang zum Netzwerk von ihrem System aus nicht funktioniert.

Im ungünstigsten Fall schafft es die nicht richtig funktionierende Karte, das ganze Netz aufzuspalten oder Geräte reihenweise zum Absturz bringen. Sie kann einen falschen, sehr alten Zeitstempel aussenden und andere Stationen auf diese Weise dazu „überreden", ihre IBSSID zu übernehmen. Dabei tritt nicht selten der Effekt auf, dass die Karten gegenseitig den Zeitstempel bis zum Überlauf (und damit bis zum Rücksetzen auf Null) hochschaukeln. Da der Zeitstempel eigentlich erst nach vielen tausend Jahren überlaufen sollte, haben viele WLAN-Programmierer nicht getestet, was in einem solchen Fall mit ihrer Software geschieht.

Aus einer einzelnen Funkzelle werden durch derartige Fehler schnell zwei oder viele Funkzellen, die nicht miteinander kommunizieren können. Bestimmte Karten halten sich dabei besonders schlau und suchen sich einen neuen, freien Funkkanal.

Treten mehrere Funkzellen mit gleichem Netzwerknamen (ESSID) aber unterschiedlicher IBSSID auf dem gleichen Kanal auf, führt das bei einigen Karten zum Absturz der Firmware – die Karten benötigen dann einen harten Reset durch das Unterbrechen der Stromversorgung (Prism2, Prism2.5, Prism3). An einen sinnvollen Betrieb eines Meshnetzwerks ist unter solchen Umständen nicht zu denken.

Treten die gleichen Probleme mit einer WLAN-Karte auch bei einem Wechsel des Betriebssystems auf, ist das ein Indiz für eine fehlerhafte Firmware. Ein Update des WLAN-Treibers ist in diesem Fall nur dann erfolgversprechend, wenn dabei auch eine neue Firmware mit installiert wird.

Die Firmware ist bei älteren WLAN-Karten in einem Flash-ROM installiert (Lucent/Orinoco von Proxim, Prism2/2.5/3 von Intersil), wird beim Initialisieren der Karte in diese geladen (Atmel At76c50X, Ralink rt73, Ralink rt2570) oder auf der CPU des PCs, in dem die Karte installiert ist, ausgeführt (Hardware-Abstraction-Layer HAL von Atheros). Bei älteren Karten muss die neue Firmware also in das Flash-ROM der WLAN-Karte geschrieben werden. Schlägt dieser Vorgang fehl, ist die Karte anschließend möglicherweise irreparabel defekt.

Leider ist nicht immer die neueste Firmwareversion auch die beste. Da hilft nur geduldiges Probieren – ein beliebtes Spiel mit älteren Prism- und Orinoco-Karten. Am Ende stellt man vielleicht doch fest, dass es einfach

keine Firmware-Version gibt, die unter allen Umständen stabil läuft. Für Hersteller ist es uninteressant, sich um Fehler in der Firmware eines schon lange Zeit nicht mehr produzierten Produkts zu kümmern.

5.4.2 Der IBSSID-Hack

Wegen der häufigen Probleme mit dem IBSSID-Splitting waren die Community-Netzwerker gezwungen, sich eine Lösung einfallen zu lassen. Durch Veränderungen im Madwifi-Treiber für den Atheros-Chipsatz unter Linux und mit etwas Trickserei bei der Ansteuerung des Broadcom-Chipsatzes ließ sich ein nicht standardkonformes Verhalten erzwingen: Die IBSSID wird fest eingestellt und ignoriert alle Versuche von anderen Ad-Hoc-Stationen, diesen über Zeitstempel zu verändern.

Damit andere Karten die fest eingestellte IBSSID übernehmen, kann auf Routern mit Broadcom-Chipsätzen mit der Freifunk-Firmware und unter OpenWRT auch der Zeitstempel manipuliert werden. Eine Karte behauptet, schon seit sehr langer Zeit in Betrieb zu sein. Damit sind alle Karten, die sich an den IEEE-Standard halten, gezwungen, die festgesetzte IBSSID zu übernehmen. Karten die in irgendeiner Weise nicht funktionieren, können sich zwar nicht mit dem Netz verbinden, aber auch keinen Schaden mehr anrichten, wenn sie ihren Zeitstempel ebenfalls erhöhen.

Der Zeitstempel kann nur einen bestimmten Maximalwert annehmen. Wird dieser überschritten, springt er zurück auf Null und beginnt wieder von vorne zu zählen. Damit dies nicht zu Problemen führt, wird der Wert auf eine Startzeit gesetzt, die einige tausend Jahre in der Vergangenheit liegt und sich deshalb kurz nach dem Rücksprung des Zählers befindet. Bei schlampig programmierter Firmware oder Treibern kann das Zurücksetzen des Zählers Probleme verursachen.

Der IBSSID-Hack hat jedoch einen gewaltigen Haken, wenn er nicht mit Sachverstand eingesetzt wird. Keinesfalls darf die Cell-ID von zwei Funknetzwerken mit unterschiedlicher ESSID auf die gleiche Adresse gesetzt werden, auch wenn sie auf unterschiedlichen Funkkanälen arbeiten. Die Freifunker verwenden beispielsweise die Cell-ID `02:CA:FF:EE:BA:BE`, da sie sich einfach merken lässt und somit auf den ersten Blick offensichtlich ist, wenn an ihr „gedreht" wurde.

Wenn nun andere Anwender für ihr kleines Mesh, das parallel zur großen Meshwolke existiert, eine neue Ad-Hoc-Zelle einrichten und die bekannte Cell-ID übernehmen, kracht es in beiden Netzwerken: Die WLAN-Karten wissen nicht mehr, mit welcher Zelle sie sich assoziieren sollen. Die Cell-ID – nicht die ESSID, wie man annehmen möchte – trennt die Funkzellen logisch voneinander. Die Zeitstempel und die ESSID dienen lediglich dazu, die IBSSID einer Funkzelle auszuhandeln. Im günstigsten Fall assoziieren sich die Karten mal mit dem einen, dann mit dem anderen Netzwerk. Bei

vielen Karten führt das jedoch zum Absturz oder zum wilden Generieren zufälliger weiterer Cell-IDs. Damit ist das Chaos perfekt.

5.5 Power over Ethernet

Am sinnvollsten ist ein Meshrouter, wenn er an einem exponierten Standort möglichst hoch und möglichst frei aufgestellt wird. Deshalb ist die Stromversorgung ein entscheidendes Problem beim Betrieb von Routern.

Eine elegante und sichere Methode, um einen WLAN-Router auf einem Dach oder an einem hohen Schornstein mit Strom zu versorgen, ist die Energieversorgung mit ungefährlicher Niedrigspannung (bis 60 Volt Gleichspannung) durch das Netzwerkkabel.

Echtes Power over Ethernet (PoE) ist ein IEEE-Standard (802.3af), der die Stromversorgung von IT-Komponenten über Entfernungen bis zu 100 Meter durch das Netzwerkkabel erlaubt. In Twisted-Pair-Kabeln für Ethernet der Kategorie 10BaseT (10 MBit/s) und 100BaseTX (100 MBit/s) befinden sich acht einzelne, voneinander isolierte Leiter. Nur vier davon werden für die Kommunikation zwischen den Netzwerkkomponenten genutzt (Grün, Grün-Weiß, Orange, Orange-Weiß). Ist ein Kabel standardkonform verschaltet, werden die Leiter mit den Farbcodes Blau und Blau-Weiß sowie die Leiter Braun und Braun-Weiß nicht benötigt. Diese vier Leiter kann man zu zwei Leitungspaaren mit dem doppelten Querschnitt kombinieren, also parallel schalten. Je ein Paar parallel geschalteter Drähte dient zum Transport der positiven oder der negativen Spannung. Der Drahtquerschnitt in Ethernetkabeln ist zwar recht dünn, aber bei hinreichend kleinen Strömen halten sich die Energieverluste in Grenzen.

Echtes Power-over-Ethernet arbeitet mit einer typischen Spannung von 48 Volt (zwischen 44 und 56 Volt) und liefert an die Verbraucher eine Leistung von mindestens 12,95 Watt. Professionelle WLAN-Router sind üblicherweise für den Betrieb per PoE vorbereitet und haben eine PoE-kompatible Spannungsaufbereitung eingebaut. Sie benötigen lediglich ein Netzteil am Stromnetz, das einen Adapter (Injector) hat. Dieser legt die PoE-Spannung auf die gemäß dem Ethernet-Standard unbeschalteten Adern 4, 5, 7 und 8.

5.5.1 Einfaches PoE im Selbstbau

Gleichspannung hat eine Polarität (Plus und Minus). Wird sie beim Anschließen der Geräte irrtümlicherweise vertauscht, können diese sofort und irreparabel beschädigt werden. Dabei können auch andere angeschlossene Geräte Schaden nehmen. Daher übernimmt die Autorin keinerlei Haftung für die folgende Anleitung, der Nachbau erfolgt auf eigene Gefahr!

Für den beliebten Linksys-Router WRT54GL oder Geräte mit ähnlich konzipierter Stromversorgung lässt sich ohne den Kauf von zusätzlicher Hardwa-

re eine einfache PoE-Lösung basteln. Diese arbeitet jedoch nicht mit 48 Volt wie das PoE nach dem IEEEE-Standard 802.3af, sondern verwendet einfach das mit dem WRT54GL gelieferte Netzteil für 12 Volt. Der Linksys-Router WRT54GL verträgt einen weiten Eingangsspannungsbereich – getestet sind 4 bis 20 Volt. Spannungsverluste führen also erst bei sehr großen Kabellängen zu Fehlfunktionen. Praktisch erprobt wurde eine Leitungslänge von rund 30 Meter.

Für diese Lösung braucht man lediglich einen Seitenschneider oder ein scharfes Messer, gutes Isolierband, Ethernetkabel passender Länge, einen kleinen Lötkolben, Elektroniklötzinn, ein Multimeter, um die Polarität der Gleichspannung zu überprüfen, und idealerweise eine Crimpzange.

Zunächst schneidet man das Kabel, das vom Steckernetzteil zum Router führt, in der Mitte durch. Beide Leitungsenden werden getrennt und beide Leiter abisoliert. Wenn man keine Crimpzange zur Verfügung hat oder das fehlerträchtige Crimpen von Ethernetsteckern vermeiden will, kann man das mitgelieferte kurze Ethernetkabel ebenfalls in der Mitte abschneiden und verarbeiten.

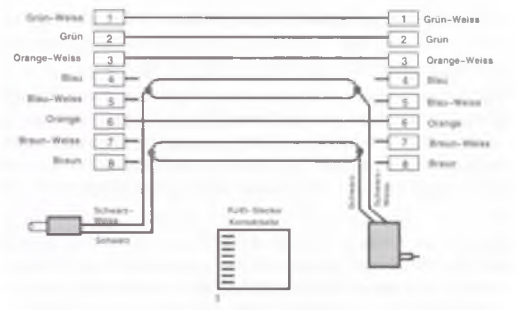


Abbildung 5.3: Schaltplan für PoE-Kabel

Am Ende des langen Ethernetkabels, das zum Router hin verlegt wird, verbindet man die Leiter Grün und Grün-Weiß und Orange und Orange-Weiß mit den Leitern mit der jeweils gleichen Farbkombination am abgeschnittenen Ende des kurzen, mit dem WRT54GL mitgelieferten Ethernetkabels (Abbildung 5.3). Auch der unisolierte Leiter in beiden Ethernetkabeln, die Abschirmung, wird zwischen beiden Enden verbunden. Die Leiter mit den Farbkombinationen Blau, Blau-Weiß, Braun und Braun-Weiß im kurzen Ethernetkabelstück werden nirgendwo angeschlossen, sondern isoliert.

Aus dem langen Kabel werden die Leiter Braun und Braun-Weiß sowie Blau und Blau-Weiß herausgeführt. Blau und Blau-Weiß werden mit dem Plusleiter des Stromversorgungskabel des Routers verbunden (Plus ist beim Linksysnetzteil mit einem weißen Strich bei schwarzem Kabel, mit einem

schwarzen Strich bei einem grauen Kabel gekennzeichnet). Der nicht gekennzeichnete Leiter des Netzteilkabels (Minus) wird mit den Leitern Braun und Braun-Weiß aus dem langen Ethernetkabel verbunden. Alle Verbindungsstellen werden nun gegeneinander mit dem Isolierband isoliert.

Am anderen Ende des langen Ethernetkabels werden diese Arbeitsschritte wiederholt. Der braune und braun-weiße Leiter des langen Ethernetkabels werden an den nicht gekennzeichneten Leiter (Minus) aus dem Netzteil angeschlossen. Die blauen und blau-weißen Leiter werden mit dem gekennzeichneten Leiter aus dem Netzteil verbunden. Am kurzen Ethernetkabel werden diese Leiter isoliert. Die Leiter Grün und Grün-Weiß und Orange und Orange-Weiß werden zwischen kurzem und langem Ethernetkabel farbgleich verbunden und isoliert. Nun kann man das Netzteil nach einer Sichtprüfung in die Steckdose stecken und mit dem Multimeter die Polarität am Hohlstecker prüfen. Beim Linksys ist der Innenkontakt des Hohlsteckers Plus, der Außenkontakt Minus.

Dazu wird das Multimeter auf die Messung von Gleichspannung bis 20 Volt eingestellt. (Darauf achten, dass die Messleitungen auf das Messen von Spannung gestöpselt sind!) Bei einem Multimeter ist das rote Messkabel Plus und das schwarze Messkabel Minus. Mit der Prüfspitze des roten Kabels berührt man nun den Innenleiter des Hohlsteckers und gleichzeitig mit der Prüfspitze des schwarzen Kabels den Außenleiter.

Schlägt bei einem analogen Multimeter der Zeiger nun in die richtige Richtung aus, wurde alles richtig angeschlossen. Ein digitales Multimeter muss die Spannung ohne Minus-Vorzeichen angezeigen. Ansonsten wurde die Spannung verpolt. Auf keinen Fall darf man in diesem Fall den Hohlstecker in den Router stecken - dieser kann unmittelbar irreparablen Schaden nehmen.

Nach erfolgreicher Prüfung kann man den Router in Betrieb nehmen und testen, ob die Verbindung zwischen PC und Router über das Netzwerkkabel klappt. Wenn auch diese Prüfung erfolgreich abgeschlossen wurde, sollte man die Lötstellen gut verpacken und gegen Zugbelastung sichern. Fertig ist das praktisch kostenlose PoE.

5.5.2 Einfache PoE-Lösungen selbst entwickeln und berechnen

Prinzipiell funktioniert die im letzten Absatz beschriebene PoE-Lösung mit allen Routern, die einen weiten Eingangsspannungsbereich haben und bei denen das Netzteil ausreichend dimensioniert ist. Es funktioniert nicht mit Routern, die mit einer niedrigen Eingangsspannung an ihrer Stromversorgungsbuchse arbeiten (5 Volt oder 3,3 Volt wie z. B. beim Buffalo WHR-G54S). In diesem Fall darf man die angegebenen Werte kaum über- oder unterschreiten. Unterschreitet die Eingangsspannung des Routers den Min-

destwert, weil das Kabel zu lang ist, wird das Gerät überhaupt nicht starten, unter Last Fehlfunktionen erzeugen oder abstürzen. Bei zu hoher Spannung wird das Gerät heiß oder geht schnell kaputt.

Leider kann man nicht einfach die Eingangsspannung auf der Netzteilseite um den Betrag erhöhen, der im Kabel verlorengeht, denn der Stromverbrauch und damit der Spannungsabfall im Netzwerkkabel ist lastabhängig. Bei ansteigendem Stromverbrauch nehmen die Spannungsverluste in der Zuleitung zu, umgekehrt nehmen sie ab. Man riskiert sonst, dass der Router bei wenig Last wegen Überspannung zerstört wird.

Die Betriebsspannung eines Routers lässt sich auf dem mitgelieferten Netzteil ablesen oder einfach mit einem Multimeter am Hohlstecker des Netzteils messen. Geräte mit einer niedrigen Spannung benötigen für PoE einen externen DC-Spannungswandler, der die PoE-Spannung heruntertransformiert. Derartige Spannungswandler und Netzteile mit den dazu passenden Spannungen können bei Elektronikversendern[7] bestellt werden. Pollin führt solche Wandler oft als Restposten zu einem Spottpreis. Fraglich ist, ob sich der Aufwand lohnt, wenn sich einfaches PoE mit einem anderen Router ohne zusätzliche Hardware realisieren lässt. Möchte man dagegen einen billigen Router über ein sehr langes Kabel speisen, ist das durchaus eine günstige Möglichkeit.

Für Router mit 12-Volt-Netzteil lässt sich PoE in der Regel problemlos realisieren. In diesen muss intern mindestens ein Spannungswandler verbaut sein, der die Eingangsspannung reduziert, denn die integrierten Bausteine der kleinen Embedded-Geräte arbeiten üblicherweise mit 3,3 Volt oder 5 Volt. Router mit 12 Volt Eingangsspannung bereiten diese entweder mit einem Linearspannungsregler oder einem getakteten Spannungswandler auf.

Linearspannungsregler wandeln die nicht benötigte Spannung über einen elektronisch geregelten Widerstand einfach in Wärme um. Arbeitet der Router intern mit 3,3 Volt, werden beim Betrieb mit 12 Volt 8,7 Volt durch den Linearspannungsregler verheizt. Entsprechend schlecht ist der Wirkungsgrad einer solchen Lösung: Reduziert ein Linearspannungswandler die Eingangsspannung von 12 auf 3,3 Volt, beträgt er 27,5 Prozent.

Einem solchen Gerät schadet es überhaupt nicht, wenn im Netzwerkkabel ein paar Volt verlorengehen – diese Energie muss nun nicht mehr im Router verheizt werden, was sich langfristig positiv auf dessen Lebensdauer auswirken dürfte. Linearspannungsregler benötigen eine Eingangsspannung, die mindestens 2–3 Volt über der Ausgangsspannung liegt. Selbst wenn einige Bauteile im Router mit 5 Volt arbeiten, darf die Eingangsspannung am Router bis auf 8 Volt absinken. Linearspannungsregler sind sehr preiswert, werden aber lobenswerterweise zunehmend von getakteten Spannungswandlern verdrängt.

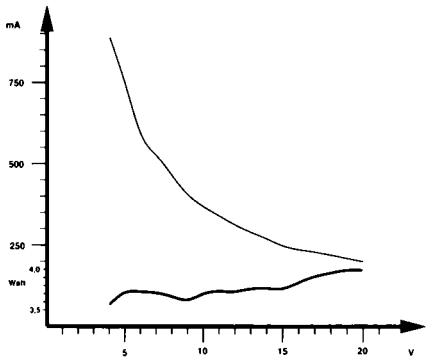
Diese transformieren Gleichspannung, ohne sie zu verheizen. Die Technik ist zwar etwas teurer, wird aber durch die Energieeinsparung und dank ge-

[7] `http://www.reichelt.de` oder `http://www.pollin.de`

ringer Probleme mit der Abwärme immer beliebter. Auch ein getakteter Abwärtswandler benötigt eine Eingangsspannung, die über der erzeugten Ausgangsspannung liegt. Hier hängt es von der Dimensionierung der Bauteile ab, wie weit die Eingangsspannung absinken darf. Als Laie kann man das nicht ohne Weiteres theoretisch ermitteln, aber ganz einfach messen. Dazu wird ein stabilisiertes Netzteil mit regelbarer Ausgangsspannung angeschlossen. Es muss an seinem Ausgang natürlich Gleichspannung abgeben, und die Polarität darf nicht versehentlich vertauscht werden. Am besten eignet sich dazu ein Labornetzgerät. Um die untere Spannungsgrenze experimentell zu ermitteln, kann unter Volllast die Versorgungsspannung reduziert werden, bis das Gerät nicht mehr stabil läuft. Theoretisch könnte es dabei kaputt gehen, praktisch dürfte dieser Fall bei modernen Bauelementen kaum auftreten, da sie normalerweise eine Strombegrenzung haben. Solange die Eingangsspannung 9 Volt nicht unterschreitet, dürfte ein Router mit 12 Volt Versorgungsspannung ohnehin kaum Probleme haben.

Die obere Spannungsgrenze muss nicht untersucht werden, wenn das Originalnetzteil des Herstellers verwendet wird. Je nach Routermodell ist es möglich, für längere Leitungen ein Netzteil mit einer etwas höheren Spannung zu wählen. Das ist vor allem interessant, wenn an einem Standort (z. B. auf einem Kirchturm) mehrere SoHO-Router mit einem Netzteil über PoE betrieben werden sollen. Die obere Spannungsgrenze kann man mit einiger Bestimmtheit von den Elektrolytkondensatoren des getakteten Spannungswandlers ablesen. Sicherheitshalber sollte man einen gewissen Betrag unter dem aufgedruckten Spannungswert bleiben. In diesem Fall muss allerdings unbedingt ein stabilisiertes Netzteil (z. B. von einem alten Notebook) verwendet werden, da ein ungeregeltes Netzteil ohne Last zuviel Spannung abgeben kann.

Abbildung 5.4: Leistungskurve des WRT54GL



Die unterschiedlichen Versionen des Linksys WRT54GL bereiten intern die Eingangsspannung mit getakteten Spannungswandlern auf. Die Autorin hat mit einem WRT54GL (Hardwarerevision 1.1) Messungen mit einem Labornetzgerät durchgeführt (Abbildung 5.4). Die Testbedingungen waren volle Rechenlast (OLSR mit 440 Routen in der Routentabelle, zwei simultane Downloads über WLAN bei 11 MBit und per Ethernetkabel mit 100 MBit, 84 Milliwatt Sendeleistung). Die Spannungsuntergrenze, bei der das Gerät gerade noch arbeitete, betrug 3,7 Volt. Darunter ging es in den *Brown-Out*, einen undefinierten Zustand aufgrund nicht ausreichender Betriebsspannung. Die reduzierte Leistungsaufnahme bei 4 Volt deutet darauf hin, dass bereits einzelne Komponenten des Routers mit zu geringer Spannung versorgt werden, aber gerade noch nicht abstürzen.

Die Messungen sind nicht ganz exakt, da der Stromverbrauch in geringem Umfang schwankt. Die angegebenen Zahlen sind die Maximalwerte. Für einen stabilen Betrieb sollte zum niedrigsten Spannungsniveau natürlich ein gewisser Sicherheitsabstand eingehalten werden. Da das Gerät intern einen Flashspeicher enthält, kann es zu Fehlern im Dateisystem kommen, wenn beim Schreiben die erforderliche Spannung zum korrekten Setzen der Speicherzellen im Flashchip nicht ausreicht.

Ist dieser Fall eingetreten, muss möglicherweise der Router aufwendig mit JTAG (siehe Abschnitt 6.7.2, Seite 163) repariert werden. 7 Volt Betriebsspannung reichen jedoch problemlos aus. Tüftler speisen ihre WRT54 sogar über zwei USB-Anschlüsse mit einem Y-Kabel mit nur 5 Volt, um die Geräte mobil mit einem Laptop einsetzen zu können. Auch bei dieser Spannung klappte bei einem Test das Schreiben im Flashspeicher noch, doch das ist „hart auf Kante genäht". So lange nicht im Flash geschrieben wird, besteht keine Gefahr. Allerdings sollte man das Gerät nicht über einen einzelnen USB-Anschluss betreiben, dazu ist dessen Leistungsabgabe von maximal 2,5 Watt zu gering.

Interessant ist auch, dass die Spannungsaufbereitung des Routers bei 9 Volt offensichtlich ihren besten Wirkungsgrad hat. Bei dieser Spannung betrug die Leistungsaufnahme unter Volllast 3,6 Watt.

Ob in einem Router eine getaktete Spannungsaufbereitung verbaut ist, wird ersichtlich, wenn man das Gerät öffnet. Bei einem Router mit einem getakteten Spannungswandler (Drosselwandler) befindet sich in der Nähe der Stromversorgungsbuchse mindestens ein relativ großer, mit Kupferdraht bewickelter Ringkern (eine Speicherdrossel). Dicht daneben findet man zwei Elektrolytkondensatoren mit einer großen Diode und dem Spannungswandler-Chip. Abbildung 5.5 zeigt das Innenleben eines WRT54GL; hier sind gleich drei getaktete Spannungswandler eingebaut, erkennbar an den drei Speicherdrosseln. Alle drei Drosselwandler teilen sich einen Elektrolytkondensator an ihrem Eingang, insgesamt sind vier Elektrolytkondensatoren in der Spannungsaufbereitung verbaut. Man erkennt auch die aufgedruckten Werte für die Spannungsfestigkeit der Elektrolytkondensatoren, die den Spannungswandlern als Puffer dienen.

Abbildung 5.5: Innenleben des WRT54GL

In Routern mit getakteten Spannungswandlern nimmt bei höherer Spannung der Stromfluss ab, wie man es aus den Messergebnissen des WRT54GL ersehen kann. Bei niedrigerer Spannung steigt der Strom an. Die Versorgungsspannung des Routers hat damit Einfluss auf die Stromstärke und somit auch auf den Spannungsabfall in der Zuleitung. Je mehr Ampere durch eine Leitung fließen, desto größer ist der Spannungsverlust. Deswegen arbeitet PoE nach 802.3a–f mit einer relativ hohen Kleinspannung von 48 Volt; auf diese Weise kann der Stromfluss kleingehalten werden.

5.5.3 Spannungsverluste auf der Leitung berechnen

Die Spannungsverluste auf einem Kabel entstehen durch den elektrischen Widerstand der Leitung und den Strom, der durch den Leiter fließt. Für die Berechnung der maximalen Leitungslänge wird zunächst eine Untergrenze für die Betriebsspannung festgelegt. Die Messergebnisse zeigen, dass der WRT54GL unter Volllast bei 7 Volt einen Strom von 0,53 Ampere benötigt.

Tabelle 5.4: Leitungswiderstände von Ethernet-Kabeln

AWG	Leiterquerschnitt in mm^2	Widerstand in Ω/m
24	0,32	0,055
23	0,26	0,068
22	0,20	0,089

Der Widerstand einer elektrischen Leitung hängt von ihrem Drahtquerschnitt ab (Tabelle 5.4). Dieser ist bei Ethernetkabeln außen aufgedruckt. Als Maßeinheit wird bei Ethernetkabeln die amerikanische Einheit AWG (*American Wire Gauge*) verwendet, wohingegen in Deutschland die Angabe des Drahtquerschnitts in Quadratmillimetern üblich ist. Zulässig für die

Herstellung von Ethernetkabeln sind die Querschnitte AWG 22–24, ein höherer AWG-Wert gibt einen dünneren Querschnitt an.

Da jeweils zwei Leiter parallel geschaltet werden, verdoppelt sich der Leiterquerschnitt. Also halbieren sich der Widerstand der Leitung und damit die Spannungsverluste. Bei AWG 22 beträgt der Widerstand also 0,0445 Ω/m bei Parallelschaltung der Drähte. Nicht vergessen darf man bei der Berechnung jedoch, dass der elektrische Strom auf einem Kabel den doppelten Weg zurücklegt, da es sich um einen Stromkreis handelt, in dem der Strom hin- und zurückfließen muss. In einem Kabel von 10 Meter Länge muss der Strom also eine Strecke von 20 Meter zurücklegen.

Der maximale Leiterwiderstand kann mit einer einfachen Formel aus dem Ohmschen Gesetz berechnet werden, wenn der zulässige Spannungsabfall und der maximal fließende Strom bekannt sind. Der zulässige Spannungsabfall beträgt in diesem Rechenbeispiel mit dem WRT54GL 5 Volt (12 Volt Netzteil minus 7 Volt untere Spannungsgrenze). Aus dem Ohmschen Gesetz ($R = U/I$) ergibt sich danach:

$$5\,V/0.53\,A = 9,43\,\Omega$$

Nun kann man leicht ausrechnen, wie lang das Kabel sein darf:

$$9,43\,\Omega / 0,0445\,\Omega/m = 212\,m$$

Das Kabel darf also unter Berücksichtigung von Hin- und Rückweg erstaunliche 106 Meter lang sein.

Anlaufstrom berücksichtigen

Bei extremen Leitungslängen mit entsprechend hohem Leitungswiderstand darf nicht vergessen werden, dass zum Start des Routers ein erhöhter Anlaufstrom und eine bestimmte Mindestspannung bereitstehen müssen, damit der Router überhaupt startet.

Die Schwierigkeit bei der Dimensionierung besteht darin, dass sich der Spannungsabfall in der Leitung und die Stromaufnahme bei einem getakteten Spannungsregler wechselseitig bedingen. Bei niedrigerer Eingangsspannung steigt die Stromaufnahme des Routers an, was zu erhöhtem Stromfluss führt. Dieser bedingt einen erhöhten Spannungsabfall in der Zuleitung. Es bildet sich ein Teufelskreislauf heraus: Erhöhter Stromfluss führt zu erhöhtem Spannungsabfall, was wiederum erhöhten Stromfluss und damit erhöhten Spannungsabfall zur Folge hat usw.

Sind die Komponenten (das gilt in erster Linie für das Netzteil) nicht ausreichend dimensioniert, bricht wegen des erhöhten Anlaufstroms die Spannung in der Versorgungsleitung beim Start so weit zusammen, dass der

Router schon beim Einschalten im Zustand des Brown-Out hängenbleibt. Der Anlaufstrom lässt sich mit einem Labornetzgerät messen, bei dem der Ausgangsstrom eingestellt werden kann. Der Maximalstrom wird am Labornetzgerät so weit reduziert, dass der Router gerade noch startet. Beim getesteten WRT54GL muss beim Start eine Spannung von 4 Volt und ein Strom von mindestens 500 Milliampere zur Verfügung stehen.

Hat man keine Möglichkeit, derartige Messungen durchzuführen, kann man als höchsten Einschaltstrom die maximale Belastbarkeit des Netzteils wählen. Legt man wie im vorigen Abschnitt eine Mindestspannung von 7 Volt an der Stromversorgungsbuchse des Routers bei einer Netzteilspannung von 12 Volt zugrunde, kommt man bei einem Strom von einem Ampere zu diesem Ergebnis:

$$5\,V / 1\,A = 5\,\Omega$$

$$5\,\Omega / 0{,}0445\,\Omega/m = 112{,}36\,m$$

Also beträgt hier die maximale Leistungslänge (Hin- und Rückweg zusammengerechnet) rund 56 Meter.

Dimensionierung des Netzteils

Die zusätzlichen Energieverluste durch die lange Versorgungsleitung müssen durch das mitgelieferte Netzteil aufgebracht werden. Dies schlägt sich in einem gesteigerten Stromfluss nieder, der durch die Messung bereits ermittelt wurde. Mit einer Leistung von 12 Watt (12 Volt, 1 Ampere) ist das mitgelieferte Steckernetzteil beim WRT54GL ausreichend überdimensioniert. Bei einem Gerät mit knapp dimensionierten Netzteil muss man dieses durch ein leistungsstärkeres stabilisiertes Netzteil ersetzen.

Kapitel 6

Mesh-Betrieb auf SoHO-Routern

Bei einigen Routern aus dem SoHO-Bereich kann die Originalfirmware des Herstellers durch eine alternative Firmware ersetzt werden. Praktisch alle im Handel erhältlichen Router stellen eine Update-Funktion für das Einspielen einer neuen Firmware zur Verfügung. Bekannt sind hier vor allem die für diese Hardware entwickelten Linux-Distributionen OpenWRT und DDWRT. Auf der Basis von OpenWRT, Version „White Russian", hat Sven-Ola Tücke eine für den Einsatz im Freifunk-Mesh spezialisierte Firmware-Distribution namens Freifunk-Firmware entwickelt.

Durch die Freifunk-Firmware wird die Installation eines für das Meshing vorbereiteten Betriebssystems in einem geeigneten Router so einfach wie ein Firmwareupdate des Herstellers. Die Freifunk-Firmware kann bei dem beliebten Router Linksys WRT54GL und ähnlichen Modellen über das Webinterface installiert werden.

6.1 Freifunk-kompatible Router

Voraussetzung für eine erfolgreiche Installation ist ein zur Freifunk-Firmware kompatibles Modell mit geeigneter Hardwarerevision. Viele Firmen verkaufen Geräte aus asiatischer OEM-Produktion unter ihrem Namen und mit firmentypischem Design, ohne diese selbst herzustellen oder zu entwickeln. Es gibt daher Geräte, deren Innenleben zu den Produkten anderer Firmen baugleich ist, da sie vom gleichen OEM-Hersteller stammen.

Leider wechselt der Inhalt der Gehäuse oft schneller als es für die Anwender wünschenswert wäre. Dabei kann sich sowohl das mitgelieferte Betriebssystem als auch das Hardwaredesign mit jeder Revisionsnummer vollständig ändern, ohne dass es äußerlich erkennbar ist, solange man nicht auf das Typenschild schaut. So war der Linksys WRT54GS bis zur Hardwarerevision 3.0 mit einem Flashspeicher von 8 MByte und einem RAM von 32 MByte vergleichsweise üppig ausgestattet. Die Hardware-Revision 4.0 enthält dagegen nur noch 4 MByte Flash und 16 MByte RAM – damit ist das Gerät zum Preis von rund 85 Euro nicht besser ausgestattet als der wesentlich billigere WRT54GL zum Preis von etwa 65 Euro. Die nachfolgenden Revisionen 5.0, 5.1 und 6.0 enthalten nur noch 2 MByte Flash und 16 MByte RAM – sie sind vollkommen uninteressant und funktionieren auch nicht mehr mit der Freifunk-Firmware.

Inzwischen scheinen einige Firmen realisiert zu haben, dass sich mit Open-Source-getriebener Hardware Geld verdienen lässt, weil das für einige Konsumenten kaufentscheidend ist. So stellte Linksys dem WRT54G nach dessen Umstellung von Linux (Hardwarerevisionen 1.0 bis 4.0) auf das proprietäre Betriebssystem VxWorks (Hardwarerevisionen 5.0 und höher) das Modell WRT54GL mit Linux zur Seite.

Eine Liste der von der Freifunk-Firmware unterstützten Router zeigt Tabelle 6.1. Dabei sind nicht alle der aufgeführten Geräte noch im Handel oder werden mit einer geeigneten Hardwarerevision angeboten.

Tabelle 6.1: Von der Freifunk-Firmware unterstützte Router

Gerät	Bemerkung
Allnet 0277	nicht mehr im Handel
Asus WL-500G	kaum noch anzutreffen
Asus WL-500G Deluxe	Mit USB 2.0, leider nicht mehr im Angebot
Asus WL-500G Premium	Mit USB 2.0, gut ausgestattet
Buffalo WHR-G54S	sehr empfehlenswertes und preiswertes Modell; Freifunk-Firmware kann nur per TFTP installiert werden

Fortsetzung:

Gerät	Bemerkung
Linksys WRT54G	alle Modelle mit Revisionsnummer 1.0 bis 4.0, neuere WRT54G-Modelle (höhere Versionsnummern) werden nicht mehr unterstützt.
Linksys WRT54GL	das empfohlene Gerät
Linksys WRT54GS	nur ältere Versionen bis Revisionsnummer 3.0 sind interessant
Linksys WAP54G	Accesspoint ohne Switch, nur 2 MByte Flash und 8 MByte RAM
Linksys WRT54G3G	eigentlich ein WRT54G mit UMTS-Card in einem Cardbus-Steckplatz
Motorola WR850G	in Deutschland selten im Handel
Siemens SE505	nicht mehr im Handel, über Ebay zu bekommen

Alle für den Betrieb mit der Freifunk-Firmware geeigneten Router basieren auf integrierten Chipsätzen der Firma Broadcom. Auch in einer Großstadt auf einem sehr hoch gelegenen Standort mit vielen Nachbarstationen läuft die WLAN-Hardware von Broadcom im Ad-Hoc-Modus stabil, wenn die IBSSID festgelegt wurde (siehe Abschnitt 5.4.2 auf Seite 115).

Üblicherweise verfügen die Geräte über eine MIPS-CPU mit 200 MHz, einen Flashspeicher von 4 MByte und 16 MByte RAM. Das sind dann auch die empfohlenen Systemanforderungen für die Freifunk-Firmware. Ein paar Geräte sind etwas üppiger mit Flashspeicher und/oder RAM und einer etwas schnelleren CPU ausgestattet. Manche Router haben auch nur 2 MByte Flash und 8 MByte RAM. Das ist eindeutig zu knapp bemessen und reicht höchstens für ein sehr minimalistisches System.

Gemeinsam ist allen Geräten, dass die WLAN-Interfaces sehr gut funktionieren und auch eine ordentliche Reichweite haben. Die Modelle, auf die jetzt näher eingegangen wird, sind aktuell im Handel erhältlich und haben mindestens 4 MByte Flash und 16 MByte RAM.

Über ein Paketmanagementsystem (iPKG, siehe Abschnitt 6.6) können in den Routern mit der Freifunk-Firmware komfortabel zusätzliche Programme installiert werden. Da sie auf der gleichen Hardwareplattform aufsetzen, ist das Angebot an Softwarepaketen für alle Modelle gleich. Nach der Installation der Freifunk-Firmware verbleiben noch knapp 2 MByte freier Speicherplatz bei Geräten mit 4 MByte Flash, die für die Installation weiterer Programme oder eigener Erweiterungen genutzt werden können.

Nicht für alle prinzipiell geeigneten und preiswerten WLAN-Geräte, die mit einer Linux-Firmware ausgestattet sind und in einem Mesh verwendet wer-

den können, gibt es ein schlüsselfertiges Firmware-Image von Freifunk. Dazu ist das Angebot einfach zu groß und verändert sich zu schnell. Hier kann die Linux-Distribution OpenWRT weiterhelfen, auf deren Grundlage die Freifunk-Firmware aufgebaut ist.

6.1.1 Linksys WRT54GL und WRT54G

Der WRT54GL ist zum Preis von rund 65 Euro das zur Zeit beliebteste Gerät für Community-Netzwerke und im Augenblick der Freifunk-Router schlechthin. Von den älteren Hardwarerevisionen des WRT54G unterscheidet sich der GL lediglich in einem weiter gesteigerten Integrationsgrad des verbauten Chipsatzes und dem Zusatzbuchstaben L für Linux auf dem Gehäuse. Bis zur Hardwarerevision 4.0 verwendete Linksys im Modell WRT54G Linux als Betriebssystem – danach wurde ein proprietäres Betriebssystem der Firma VxWorks und ein anderer Bootloader verwendet. Gleichzeitig halbierte Linksys die Größe des Flashspeichers von 4 MByte auf 2 MByte. Auch das RAM schrumpfte dabei von 16 MByte auf 8 MByte.

Nach einiger Zeit waren engagierte Bastler in der Lage, auch in diesem Gerät Linux zu installieren – doch die aktuellen Revisionen des WRT54G sind wegen der mageren Ausstattung die Mühe kaum wert. Zudem wird die Sparversion des WRT54G im Handel mitunter zu einem höheren Preis angeboten als der in jeder Hinsicht bessere WRT54GL.

Die Installation der Freifunk-Firmware lässt sich ganz einfach über das Webinterface bewerkstelligen. Der Router verfügt über einen 4-Port-Switch und einen Uplink-Port für den Anschluss eines DSL-Modems oder die Verbindung zum bestehenden Ethernet. Auf der Funkseite kann der Router mit zwei Antennen und einem eingebauten Diversity-Chip aufwarten, der beim Empfang von Signalen überprüft, über welche Antenne eine andere Station am besten zu empfangen ist. Dabei beherrscht der WRT54GL auf seinen Antennen so genanntes True-Diversity – das Gerät kann sich nicht nur beim Empfang, sondern auch beim Senden entscheiden, über welche Antenne es eine bestimmte Station am besten erreichen kann.

Die Sendeleistung lässt sich auf dem Webinterface der Freifunk-Firmware zwischen 1 und 84 mW einstellen. Die Empfangsleistung dieses Gerätetyps ist sehr gut, selbst mit den mitgelieferten Stummelantennen sind zwei Kilometer bei freier Sicht und freier Fresnel-Zone bei niedriger Datenrate problemlos möglich (siehe Seite 184 im Kapitel 7.6.3). Das Gerät verträgt einen weiten Eingangsspannungsbereich von rund 8 bis 20 Volt an seiner Kleinspannungsbuchse. Einfache Power-over-Ethernetlösungen lassen sich damit leicht realisieren, wie im Kapitel 5 beschrieben.

Bis vor kurzem war das mitgelieferte ungeregelte Netzteil von der billigen und stromfressenden Sorte. Inzwischen wird der WRT54GL mit einem kleinen und sparsamen, getakteten Schaltnetzteil geliefert. Beide Netzteilvari-

anten sind mit einer Abgabeleistung von 12 Watt für den Betrieb des kleinen Routers reichlich überdimensioniert. Der Accesspoint verbraucht bei 12 Volt Eingangsspannung typischerweise nur etwa 0,3 Ampere Strom.

Auf der Webseite von OpenWRT finden sich auch zahlreiche Vorschläge, wie der Router modifiziert werden kann. Engagierte Bastler löten sogar einen SD-Card-Slot ein und übertakten das kleine Gerät, wie das bei den PCs von Gamern üblich ist.

Unter `http://212.222.128.68/sven-ola/ipkg/_g+gl/` findet man das zu diesem Routermodell und alten Versionen des WRT54G passende Firmware-Image.

6.1.2 Buffalo WHR-G54S

Der Buffalo WHR-G54S ist zu einem Preis von etwa 40 Euro bedeutend preiswerter als der Linksys WRT54GL – dabei steht er ihm nicht unbedingt nach. Prozessorleistung, RAM-Ausstattung und Flashspeicher sind gleich. Wie der WRT54GL verfügt das Gerät über einen eingebauten 4-Port-Switch und einen sogenannten WAN-Port für den Uplink (üblicherweise über ein DSL-Modem oder Ethernet).

Etwas schwieriger als beim Linksys ist der Installationsprozess der Freifunk-Firmware. Versucht man es über das Webinterface der Original-Firmware, lehnt der Router das Freifunk-Image als ungültig ab. In diesem Fall kann man auf das Einspielen des Firmwareimages via TFTP ausweichen. Ist die Freifunk-Firmware erst einmal installiert, können weitere Firmware-Updates problemlos über das Webinterface vorgenommen werden.

An einigen Stellen hat Buffalo an der Hardware gespart: Es gibt nur eine externe Antenne und eine billige Behelfsantenne im Gehäuse. Das ist aber beim Betrieb mit einer leistungsfähigen Außenantenne nicht unbedingt ein Nachteil. Unter manchen Gesichtspunkten ist der Buffalo WHR-G54S sogar besser als der WRT54GL. Bei gleicher Funktionalität ist das Gerät wesentlich kleiner und sieht zumindest subjektiv besser aus. Ebenso wie der WRT54GL arbeitet der Buffalo stabil im Ad-Hoc-Modus und hat eine sehr gute Reichweite.

Der WHR-G54S verbraucht wenig Strom aus der Steckdose, da er mit einem effizienten und modernen, getakteten Steckernetzteil geliefert wird. Auf eine aufwendige Spannungsaufbereitung im Gehäuse des Routers wie beim WRT54GL wurde bei diesem Modell verzichtet. Der Gleichspannungseingang des WHR-G54S braucht eine gut geregelte Kleinspannung von 3,3 Volt. Da die Spannung sehr viel kleiner ist, ist die Stromstärke höher als beispielsweise beim WRT54GL. Ein einfaches Versorgen des Routers per selbstgebasteltem PoE ohne zusätzlichen getakteten Spannungswandler ist deshalb beim WHR-G54S nicht möglich. Für den Betrieb eines Buffalo sollte man deshalb eine Steckdose in der Nähe haben.

Damit eignet sich das Gerät nicht so gut für den Betrieb auf einem Mast im Freien. Da Kabel im Freien verwittern, ist Netzstrom auf einem Mast ein Risikofaktor. Für den Betrieb in der Wohnung oder am Fensterbrett ist der „Büffel" dagegen sehr gut geeignet. Wenn man eine PoE-Lösung für weniger als 20 Euro Materialkosten bauen kann und will, ist das kleine Gerät aber auch für Außeneinsätze interessant.

Unter `http://212.222.128.68/sven-ola/ipkg/_trx/` findet man die passende Firmware.

6.1.3 Asus WL500G Premium

Der Asus WL500G Premium ist ein mit zusätzlichen Funktionen ausgestatteter WLAN-Router zum Preis von etwa 90 Euro. Das Gerät besitzt einen Prozessor mit 266 MHz, 8 MByte Flash und 32 MByte RAM. Das hervorstechendste Merkmal ist das Vorhandensein von zwei schnellen USB-2.0-Anschlüssen. Damit kann man an das Gerät externe Festplatten, Speichersticks, zusätzliche WLAN-Karten, Drucker oder eine Soundkarte anschließen. Der WL500G Premium ist also ein Router mit vielen Möglichkeiten, der jedoch nur über einen externen Antennenanschluss verfügt. Das Gehäuse ist für einen SoHO-Router relativ groß.

Der Router eignet sich für den Betrieb über Power over Ethernet nur, wenn ein zusätzlicher externer Spannungswandler eingesetzt wird, da er mit 5 Volt Eingangsspannung arbeitet. Zusätzliche Komponenten, die ihre Energie über einen der beiden USB-Anschlüsse beziehen, treiben den Stromverbrauch des Routers natürlich in die Höhe. Für die Montage an einem Mast ist das Gerät eigentlich viel zu schade.

Dieser Router arbeitet mit dem gleichen Firmware-Image wie der Buffalo WHR-G54S. Die Installation der Firmware kann ebenso wie beim Buffalo WHR-G54S nicht über das Webinterface erfolgen. Durch Drücken des Reset-Knopfes kann man den TFTP-Server des Bootloaders aktivieren, der unter der IP 192.168.1.1 zu erreichen ist. Nun kann per TFTP-Upload das Firmware-Image mit einem TFTP-Client installiert werden. Ist die Installation abgeschlossen, startet der Asus nicht automatisch neu. Nachdem man fünf Minuten gewartet hat, kann man das Gerät vom Strom trennen. Beim nächsten Start bootet das Gerät dann in die Freifunk-Firmware.

6.2 Installation der Freifunk-Firmware

Selbstverständlich erlischt bei allen im Folgenden beschriebenen Eingriffen die Herstellergarantie und man handelt auf eigene Gefahr.

Beim Installieren eines Firmware-Image muss dafür Sorge getragen werden, dass der Flashprozess nicht vorzeitig unterbrochen wird (jemand zieht aus

Versehen den Mehrfachstecker aus der Dose, eine Steckdose hat einen Wackelkontakt, es wird nicht gewartet, bis der Flashprozess abgeschlossen ist). Dabei ist es gleichgültig, ob die Firmware per Webinterface oder per TFTP-Upload installiert wird. Schlägt die Installation der Firmware wegen einer vorzeitigen Unterbrechung fehl, hat man das Gerät möglicherweise in einen hilflos blinkenden Briefbeschwerer aus thermoplastischem Kunststoff verwandelt – „gebrickt" (von engl. Brick = Ziegelstein) oder „zerflasht", wie es im Jargon heißt. Bei einem Stromausfall während des Flashvorgangs kann es passieren, dass das Gerät auch den Bootloader beschädigt, während die Spannung zusammenbricht[1]. Außerdem sollte man überprüfen, ob die zu installierende Firmwaredatei vollständig und unbeschädigt ist und tatsächlich zu dem Router passt.

Im Downloadverzeichnis der Freifunk-Firmware befindet sich eine Textdatei mit der Bezeichnung `md5sum.asc`, diese enthält die Prüfsummen der Firmware-Images, die einen Test auf Unversehrheit ermöglichen. Um die Prüfsumme der heruntergeladenen Dateien zu ermitteln, steht unter Linux das Programm `md5sum` zur Verfügung. Windows-Anwender können das Programm `md5sum.exe` verwenden, das ebenfalls auf dem Server liegt.

Beim ersten Flashen der Firmware sollte man den PC unbedingt per Ethernetkabel mit einem Ethernetport im Switch des Routers verbinden und nicht über WLAN flashen.

6.2.1 Firmware-Installation auf Linksys WRT54GL

Die Installation ist bei diesem Gerät sehr leicht und erfolgt am einfachsten über das Webinterface des Routers. Router und Arbeitsplatzrechner werden mit einem Patchkabel verbunden. Der WRT54G/GL hat fünf Ethernetschnittstellen, eine WAN-Buchse und vier LAN-Buchsen. In eine der vier Ethernetbuchsen (Ports) wird das Patchkabel gesteckt.

Im Neuzustand verwendet der Router die IP-Adresse 192.168.1.1 und weist einem per Ethernetkabel angeschlossenen Computer per DHCP eine IP-Adresse zu, wenn der PC entsprechend eingerichtet ist. Ansonsten kann man auch der Netzwerkkarte manuell eine IP-Adresse zwischen 192.168.1.2 und 192.168.1.254 statisch zuweisen. Eine eventuell vorhandene WLAN-Karte im PC sollte für die Firmwareinstallation sicherheitshalber deaktiviert werden, damit man nicht aus Versehen den Router per WLAN zu flashen versucht – auch hier gibt der Router per DHCP Adressen aus.

Jetzt kann man mit einem Webbrowser die URL `http://192.168.1.1` öffnen. War die Netzwerkkonfiguration erfolgreich, befindet man sich nun im Webinterface des Routers. Sollte die Verbindung partout nicht klappen, hilft

[1] Führt man das Flashen in einem Schwellen- oder Entwicklungsland aus, tut man also gut daran, das betreffende Gerät direkt aus einer geladenen Batterie mit der passenden Spannung zu speisen, da man nie weiß, wann das nächste Mal der Strom ausfällt.

vielleicht die Bedienungsanleitung auf der mitgelieferten CD-ROM, um die Standard-IP der WRTs zu überprüfen. Der Router fragt nun nach einem Benutzernamen und Passwort. Hier muss lediglich das Default-Passwort `admin` eingegeben werden, der Benutzername wird leer gelassen.

Abbildung 6.1: Original-Webinterface des Linksys-Routers

Nun wählt man im Webinterface rechts oben die Schaltfläche **Verwaltung** aus (Abbildung 6.1), woraufhin darunter die Schaltfläche **Firmware aktualisieren** sichtbar wird. Diesen Menüpunkt ruft man auf (Abbildung 6.2).

Abbildung 6.2: Firmware aktualisieren

Wird nun die Schaltfläche **Durchsuchen** ausgewählt, erscheint ein Auswahlmenü im Browser, mit dem man in den Verzeichnissen auf dem PC navigieren kann. Damit wählt man die passende Firmware-Datei aus. Ein Klick auf die Schaltfläche **Aktualisieren** startet den Flashvorgang.

Wichtig: Linksys hat in dieses Menü einen Fortschrittsbalken eingebaut. Dessen Anzeige (Abbildung 6.3) ist irreführend und hat absolut nichts mit dem tatsächlichen Fortschritt des Programmiervorganges zu tun.

Abbildung 6.3: Der Fortschrittsbalken ist irreführend

Die Anzeige im Browser meldet nach kurzer Zeit, dass der Prozess erfolgreich abgeschlossen wurde, und lädt eine Statusseite mit der Meldung **Aktualisierung war erfolgreich**. Diese Meldungen werden über ein vom Webbrowser des PCs ausgeführtes CGI-Skript nach dem Ablauf eines vorgegebenen Zeitintervalls erzeugt.

Auf keinen Fall darf man dieser Meldung (Abbildung 6.4) glauben und den Router jetzt ausstecken, um ihn neu zu starten! Klickt man auf **Fortfahren** erscheint eine Fehlermeldung. Zieht man in den nächsten zwei bis drei Minuten entnervt den Stecker, weil scheinbar nichts mehr geschieht, ist die Firmware defekt und der Router startet nicht mehr.

Abbildung 6.4: Noch eine Meldung, der man nicht glauben darf

Man muss etwa fünf Minuten warten, bis der Programmierprozess abgeschlossen ist. Ist das Flashen beendet, kann man die Begrüßungsseite der Freifunk-Firmware unter der URL `http://192.168.1.1` aufrufen. Erst jetzt darf der Router von der Stromversorgung getrennt werden.

Abbildung 6.5: Das neue Freifunk-Interface

6.2.2 Firmware-Installation per TFTP

Nur bei wenigen Geräten lässt sich die Freifunk-Firmware über das Webinterface installieren. Bei den meisten Routermodellen ist man auf die Installation über TFTP angewiesen.

TFTP steht für Trivial File Transfer Protocol und ist ein sehr einfach gehaltenes Dateitransferprotokoll, das bei vielen Embedded Systemen verwendet wird, um über den Bootloader (vergleichbar mit dem BIOS von PCs) durch eine Netzwerkverbindung Dateien in die Geräte zu übertragen. Wie das bekannte FTP-Protokoll benötigt TFTP dazu einen Client und einen Server. Ob sich im Router ein TFTP-Client oder ein Server befindet, hängt von der Implementierung des Bootloaders ab. Bei den zur Freifunk-Firmware kompatiblen Routern kommt üblicherweise der Bootloader CFE zum Einsatz, der einen TFTP-Server enthält. Um ein Firmware-Image per TFTP zu installieren, muss also ein TFTP-Client auf dem Arbeitsplatzrechner vorhanden sein. Ein Passwort benötigt man für TFTP nicht.

Prinzipiell kann man bei allen Modellen eine Firmware über TFTP installieren, doch nicht bei allen Geräten ist der TFTP-Server ab Werk eingeschaltet. Beim WRT54GL ist er in der Fabrikeinstellung deaktiviert.

Beim Asus WL500G Premium lässt sich der TFTP-Server über die Restore-Taste an der Rückseite des Gerätes aktivieren. Die Taste wird gedrückt, bevor der Router an die Stromversorgung angeschlossen wird, und danach ein paar Sekunden gedrückt gehalten. Wenn die Power-LED langsam blinkt, ist der Router bereit für den TFTP-Upload. Eine Besonderheit des Asus WL500G Premium ist, dass der TFTP-Server die gleiche Ethernet-IP-Adresse benutzt, die von den Anwendern für den Betrieb konfiguriert wurde. Im Auslieferungszustand ist es die IP-Adresse 192.168.1.1.

Nicht bei allen Modellen der Asus-Serie WL500G ist die Vorgehensweise so einfach wie beim WL500G Premium. Beim nicht mehr im Handel erhältlichen Modell Asus WL500G (ohne den Zusatz Deluxe oder Premium) gestaltet sich die Installation per TFTP wesentlich komplizierter. Für Informationen über die Installation per TFTP bei den verschiedenen Routermodellen ist das OpenWRT-Wiki[2] eine sehr gute Informationsquelle, die detailliert auf die einzelnen Modellen eingeht. Über den Link in der Status-Spalte der Hardware-Tabelle auf der Seite **TableOfHardware** gelangt man zu Informationen für das spezifische Modell.

Beim Buffalo WHR-G54S ist der TFTP-Server werksseitig eingeschaltet. Unmittelbar nach dem Einschalten des Routers leuchten bei diesem Gerät alle grünen Kontroll-LEDs an den Netzwerkbuchsen. Sobald sie erlöschen, ist der Server in Betrieb. Nun hat man etwa drei Sekunden Zeit, um den Transfer zu initiieren. Nach der vollständigen Übertragung der Firmware-Datei schreibt der Bootloader das Image selbständig in den Flashspeicher des Routers.

[2] `http://wiki.openwrt.org/`

Ebenso einfach wie das TFTP-Protokoll sind auch die meisten dazugehörigen Programme gehalten. Oft sind besonders primitive TFTP-Clients bereits auf Desktop-Rechnern oder Laptops installiert. Mit diesen Programmen ist das Flashen ein hektisches und fehlerträchtiges Glücksspiel. Die Schwierigkeit besteht darin, innerhalb der kurzen Zeit, die der TFTP-Server des Bootloaders wartet, eine Verbindung herzustellen, den TFTP-Befehl `put Dateiname` fehlerfrei einzugeben und die Enter-Taste zu drücken. Das ist zwar spannend und aufregend, klappt aber nur selten auf Anhieb.

Besser und einfacher lässt sich dieser Prozess bewerkstelligen, wenn ein TFTP-Client eingesetzt wird, der sich skripten lässt. Ein besserer TFTP-Client übernimmt beim Start die passenden Anweisungen als Kommandozeilenoptionen und führt diese Kommandos entsprechend selbständig aus. Ein passendes Programm ist `atftp` für Linux. Die Syntax zum Flashen eines Routers lautet:

```
atftp -p -l firmware-datei.xxx   IP-Adresse-Router
```

Damit ist die Installation per TFTP nicht mehr schwierig: Die IP-Adresse des Routers wird der Betriebsanleitung entnommen. Beim Buffalo WHR-G54S ist sie in der Grundeinstellung 192.168.11.1. Die Ethernetkarte des Arbeitsplatzrechners wird mit einem der vier Ethernetanschlüsse des Routers durch ein Patchkabel verbunden. Manuell wird die Ethernetkarte des PC für eine passende statische IP-Adresse konfiguriert, z. B. 192.168.11.2.

Anschließend tippt man den passenden Kommandozeilenaufruf im Terminal ein, drückt aber noch nicht die Eingabetaste. Der Router wird an die Stromversorgung angeschlossen. Sobald die Ethernet-LEDs ausgehen, hat man drei Sekunden Zeit, um bequem die Eingabetaste zu drücken. Nun startet der TFTP-Client den Upload der Firmwaredatei in den Router; dieser Transfer ist nach wenigen Sekunden beendet. Anschließend startet der CFE den eigentlichen Programmierprozess, der nicht unterbrochen werden darf. Nach etwa fünf Minuten ist der Installationsprozess abgeschlossen. Unter der URL des Routers sollte in einem Browser die Weboberfläche der Freifunk-Firmware erscheinen.

Für Windows-Anwender gibt es das Freeware-Programm `tftpd32` auf der Homepage von Philippe Jounin[3].

6.3 Konfiguration per Webinterface

Der bevorzugte Weg, die frisch installierte Firmware zu konfigurieren, ist das Webinterface. Der Hauptzweck der Freifunk-Firmware ist es, die Einrichtung eines Meshrouters für die Anwender möglichst einfach und kom-

[3] `http://tftpd32.jounin.net`

fortabel zu gestalten. Wie bei OpenWRT kann die Konfiguration des Routers aber auch auf der Kommandozeile per SSH (Secure Shell) erfolgen. Das Kommandozeileninterface ist viel mächtiger als die Weboberfläche, aber auch schwerer zu bedienen. Im Webinterface können nur diejenigen Schritte ausgeführt werden, für die es Checkboxen und vorbereitete Menüs gibt. Das sind bei aktuellen Versionen der Firmware schon eine ganze Menge. Mit dem Kommandozeileninterface kommt man dagegen auch an die ungewöhnlichsten Einstellmöglichkeiten heran.

Das Webinterface der Freifunk-Firmware hat einen öffentlichen Bereich und einen administrativen Bereich, der durch ein Passwort geschützt ist. Der Administrationsbereich ist über die Schaltfläche **Verwalten** zugänglich.

6.3.1 Sicher anmelden über SSH

Im Gegensatz zur Originalfirmware muss man nun beim Anmeldeprozess im Webinterface den Benutzernamen `root` und das Passwort `admin` angeben. Das Passwort wird dabei unverschlüsselt übertragen.

Abbildung 6.6: Unverschlüsselte Passwortübertragung im Freifunk-Interface

Die Firmware verzichtet aus Platzgründen auf SSL-Verschlüsselung für das Webinterface. Es ist also nicht möglich, sich über eine URL mit vorgestelltem `https://` mit dem Gerät zu verbinden. Das ist kein Problem, solange der Rechner, von dem aus die Wartungsarbeiten erfolgen, direkt mit dem Router per Kabel verbunden ist und keine anderen Nutzer zu diesem Netzwerksegment Zugang haben. Wenn man sich später über eine unverschlüsselte Funkverbindung oder über ein unsicheres LAN mit dem Router verbindet, ist es jedoch ein Kinderspiel, die Passwörter mitzulesen.

Dem lässt sich mit einer verschlüsselten Datenverbindung (SSH-Tunnel) zu dem Gerät ein Riegel vorschieben. Dafür gibt es unter Linux/Unix das Programm `ssh`, unter Windows kann man die Freeware PuTTY verwenden. Dazu führt man auf dem Arbeitsplatzrechner auf der Kommandozeile den Befehl aus, auf den die Login-Maske (Bild 6.6) schon hinweist. Unter Linux:

```
ssh root@router-ip -L 8080:localhost:80
```

und unter Windows:

```
C:Programme\Putty\PuTTY -L 8080:localhost:80 root@router-ip
```

SSH fragt nun nach dem Administratorpasswort des Routers. Steht die SSH-Verbindung, werden die Ein- und Ausgaben des Webservers durch den verschlüsselten Tunnel zwischen Router und PC transportiert und können so nicht mehr abgehört werden. Nun kann man auf dem PC die lokale Adresse `http://localhost:8080` öffnen. Es ist nicht zwingend der Port 8080 nötig, es eignet sich jeder freie Port auf dem lokalen PC.

6.3.2 Konfigurationsschritte

Der mittlere Teil der Webseite zeigt eine Aufgabenliste, die Schritt für Schritt abgearbeitet werden sollte. Dazu öffnet man die Menüpunkte auf der linken Seite nacheinander und konfiguriert sie. Es ist nicht notwendig, nach jedem Konfigurationsschritt einen Neustart des Systems durchzuführen, auch wenn bei fast allen Menüpunkten nach der Bearbeitung die Meldung erscheint, dass die Veränderungen erst nach dem Neustart wirksam werden. Es genügt ein einziger Neustart nach dem Beenden der Konfiguration.

Viele Menüeinträge dürften am Anfang Fragen aufwerfen. Dafür ist eine interaktive Hilfe vorgesehen. Klickt man mit der Maus in den betreffenden Einstellpunkt, kann mit (F1) eine Erläuterung aufgerufen werden.

Abbildung 6.7: Zu konfigurierende Menüpunkte

6.3.3 Kennwort und Kontaktinfos

Es ist empfehlenswert, das Administrator-Passwort unmittelbar zu ändern, da man sonst leicht das Opfer einer plumpen Attacke werden kann. Angreifer könnten unter anderem zu den Zugangsdaten für einen DSL-Anschluss

kommen, falls der Router als Einwahlpunkt für einen Internetzugang dient. Gerade an diesem Punkt empfiehlt es sich, auf eine sichere Übertragung zwischen Arbeitsrechner und Router zu achten. Leicht zu erratende Passwörter verbieten sich von selbst.

Angaben unter dem Menüpunkt **Kontaktinfos** sind für den Betrieb des Systems nicht unbedingt erforderlich, gehören aber unter sozialem Gesichtspunkt zum guten Ton. Zumindest eine E-Mail-Adresse sowie Angaben zum Standort sollte man hinterlegen. Kontaktinfos sind vor allem hilfreich, wenn sich die Betreiber der Router untereinander koordinieren wollen. Bei einem Netzwerk, das auf gegenseitiger Hilfe beruht, sicherlich kein falscher Ansatz. So können sich leicht verschiedene Parteien zusammenfinden, um beispielsweise einen gemeinsamen schnellen Gateway einzurichten oder Linkstrecken auszubauen. Ehe man es sich versieht, findet man sich mit neuen interessanten Zeitgenossen und Zeitgenossinnen am Stammtisch wieder und redet begeistert über das tolle Hobby.

6.3.4 System

Hier wird es nun endgültig technisch. Vor allem geht es unter **System** um die Einrichtung eines eigenen Domain-Namen-Systems (DNS) mit eigener Subdomain, Rechnernamen und eigenem Domain-Name-Server. Computer im lokalen Netzwerk können als DNS-Server diesen Router angeben, dieser führt stellvertretend für sie Anfagen nach Domainnamen im Internet aus und löst Domain-Namen auf. Tippt man beispielsweise im Browser die Adresse `www.freifunk.net` ein, muss der eigene Arbeitsplatzrechner herausfinden, unter welcher IP-Adresse (z. B. 192.109.42.84) diese Webseite abgerufen werden kann.

Der Vorteil eines eigenen Domain-Name-Servers liegt darin, dass auch lokale Maschinen über nur lokal gültige Domainnamen ansprechbar sind statt lediglich über IP-Nummern. Es ist also durchaus empfehlenswert, die Schaltfläche **Starte DNS/DHCP-Server: Einschalten** zu wählen.

Der Eintrag **DNS-Server** gibt den Server an, den der Router seinerseits nach ihm unbekannten Domainnamen befragt. Hier sollte ein zuverlässiger DNS-Server angegeben werden, da mit dessen Erreichbarkeit die Benutzbarkeit von Internetdiensten steht und fällt. Listen von DNS-Servern findet man im Netz[4]. Ein paar davon sollte man sich notieren, bevor man mit der Konfiguration beginnt und die DNS-Auflösung vielleicht nicht funktioniert.

Ist der DNS-Server nicht erreichbar, können Anwender keine Webseiten mehr abrufen, obwohl Internetzugang besteht, da die Namensauflösung nicht mehr funktioniert. Ist dieser Router ein Gateway, das beispielsweise an einem DSL-Modem hängt, sollte man den DNS-Server des Internet Service Providers angeben – falls dieser etwas taugt und nicht dauernd ausfällt.

[4] Beispielsweise unter `http://dns.stanar.de/`

Die beiden nächsten Felder zu aktivieren ist dagegen für unbedarfte Anwender nicht empfehlenswert. OpenWRT hat ein neues Overlay-Dateisystem namens Mini-Foo implementiert, das sich bislang noch im Experimentierstadium befindet. Der einzige Grund es einzusetzen wäre etwas Platzersparnis im Flash-Speicher, den man sich aber mit vermutlich reduzierter Datensicherheit erkauft.

Auch der nächste Punkt **Netzwerk-Startmeldungen**, der den Router bei einem Neustart dazu veranlasst, Benachrichtigung über den Neustart an das Netzwerk zu verschicken, ist nur für jene interessant, die diese Funktion unbedingt haben wollen.

Die Wahl der Zeitzone bedarf wohl kaum einer Erläuterung. Die Auswahl des Landes, in dem der Router betrieben werden soll, dürfte ebenfalls kaum Fragen aufwerfen, solange man nicht gegen die jeweiligen Regulierungsbedingungen verstoßen möchte. Es gibt Länder, in denen beispielsweise der Betrieb von WLAN auf den Funkkanälen 1-14 erlaubt ist, in Deutschland sind es 1-13. Wählt man das betreffende Land, erlaubt das Webinterface nur die landesüblichen Einstellungen.

6.3.5 OLSR

In diesem Menü wird das Verhalten des Olsrd konfiguriert. Es ist naturgemäß etwas umfangreicher und besonders erklärungsbedürftig.

Olsr-Filter

Hier kann man die IP-Adressen von benachbarten Meshknoten eingeben, deren IP-Pakete ignoriert werden. Das ist dann nützlich, wenn es besser ist, diese Knoten nicht über eine direkten, drahtlosen Funklink zu erreichen, sondern über Kabel. Hat man beispielsweise zwei OLSR-Router mit einem Kabel verbunden, sehen sich beide jeweils über ihre Ethernetschnittstelle und über ihre WLAN-Karten. Es wäre ziemlich ungünstig, wenn beide anfangen, sich Daten über ihre Antennen weiterzuleiten, statt die viel schnellere Kabelverbindung zu verwenden. In diesem Fall wird bei beiden Routern jeweils die OLSR-Adresse des WLAN-Interface des Nachbarn gefiltert.

DMZ-Umleitung

Normalerweise können Rechner, die an einem LAN-Port des Routers angeschlossen sind, nicht drahtlos vom Mesh aus erreicht werden, da sie durch die Router-Firewall vor Zugriffen aus dem Mesh geschützt sind. Über einen Eintrag in diesem Eingabefeld kann man den Zugriff aus dem Mesh auf einen Rechner am LAN-Port ermöglichen. Dazu wird eine zusätzliche IP-Adresse aus dem Adressbereich des Mesh annonciert, diese wird umge-

leitet auf eine interne LAN-IP-Adresse. Ein Beispiel: Das Mesh verwendet den Adressbereich 10.140.0.0/16, die Adresse 10.140.133.41 ist noch frei. Ein Rechner am LAN-Port mit der Adresse 172.29.178.34 soll aus dem Mesh zugänglich gemacht werden. In diesem Fall lautet die Syntax in dem Eingabefeld:

```
10.140.133.41:172.29.178.34;
```

OLSR-DHCP

Durch dieses Eingabefeld kann ein Meshrouter als Internetzugangspunkt für lokale WLAN-Stationen eingerichtet werden, die nicht das OLSR-Protokoll sprechen. Diese Clients erhalten per DHCP-Protokoll automatisch eine IP-Adresse, Netzmaske, Broadcastadresse, Gateway-Adresse und Domain-Name-Server-Adresse von diesem Meshrouter zugewiesen. Dazu muss auf den Clients kein OLSR-Daemon installiert sein. Sie werden durch eine Firewall gegen das Mesh geschützt und können zwar über den Ad-Hoc-Router alle Knoten im Mesh erreichen und auch den Internetzugang über das Mesh benutzen, sind aber vom Mesh aus nicht erreichbar.

Man sollte darauf achten, dass die DHCP-Adressen, die von diesem Meshrouter herausgegeben werden, nicht auch von anderen Routern im Mesh annonciert werden, da diese sonst von den lokalen Stationen nicht erreichbar sind. Bekannt sind allerdings die bereits hinlänglich beschriebenen Treiberprobleme im Ad-Hoc-Modus von WiFi im Vergleich zu einem Accesspoint im Infrastrukturmodus. Die Syntax lautet:

```
10.134.111.200/29,255.255.255.248
```

HNA4

Der Router teilt dem Mesh über Host-Network-Announcement mit, dass er ein Gateway zu dem/den announcierten Netzwerk/Netzwerken ist. Die Administratoren der unterschiedlichen Mesh-Knoten müssen sich unbedingt absprechen, wer welche Subnetzwerkadressen benutzt und bekannt gibt. Annoncieren mehrere Meshrouter dieselben Netzwerke, führt das zu Kollisionen, da dieselben Adressen mehrfach verwendet werden.

Vor allem die bekannten Klasse-C-Netzwerke 192.168.0.0 und 192.168.1.0, die praktisch immer und überall verwendet werden, sollte man unbedingt vermeiden und lieber etwas ganz Ungewöhnliches und Exotisches nehmen. Auch Routen zu einem einzelnen Knoten sind möglich. Mehrere Einträge können durch ein Semikolon getrennt werden (siehe hierzu auch den Abschnitt 6.5.2). Zwar kann auch eine Route ins Internet (0.0.0.0/0) über HNA4 angekündigt werden, da dies aber zu den bereits beschriebenen Schwarzen

Löchern führen kann, sei davon abgeraten. Stattdessen sollte für Internet-Gateways das dynamische Gateway-Plugin von OLSR verwendet werden.

Beispiel für eine Hostroute und eine Route zu einem Klasse-C-Netzwerk:

```
10.172.119.234/32;172.31.191.0/24
```

Broadcast IPV4

Normalerweise wird bei OLSR-Broadcasts die konfigurierte Broadcastadresse verwendet. Diese kann aber manuell für alle OLSR-Broadcasts unabhängig von ihrer IP-Konfiguration gesetzt werden. Typisch und sinnvoll wäre 255.255.255.255. Tatsächlich macht aber niemand davon Gebrauch. Dieses Feld sollte also leer bleiben.

OLSR Tempo

Die Bezeichnung ist etwas irreführend. Es handelt sich um das Emmissionsintervall der OLSR-Hello-Nachrichten. Die Freifunk-Firmware schlägt zwei Sekunden für kleinere Netzwerke vor, für größere Netzwerke mit einigen hundert Meshroutern wie in Berlin oder Leipzig ist der empfohlene Wert fünf Sekunden.

Bereitschaftswert

Dies ist die OLSR-Willingness, also die Bereitschaft, für andere Knoten Daten weiterzurouten. Der Maximalwert ist 7. Da ein Meshrouter üblicherweise am Stromnetz hängt, sollte die Eintragung 7 lauten – volle Bereitschaft. Nur bei batteriebetriebenen Systemen macht eine Reduzierung des Wertes Sinn, um Strom zu sparen. Auf Laptops und PDAs kann OLSR diesen Wert dynamisch anhand des Batteriestands anpassen, wenn kein Bereitschaftswert vorgegeben wird. Derartige Stromsparmechanismen gibt es aber in einem SoHO-Router nicht.

QOS-Protokoll (ETX)

Hier gibt es nur eine Auswahl: Einschalten! Alles andere ist Nonsense, sofern man nicht selbst herausfinden möchte, wie unsinnig Hysterese ist.

OLSR LQ-Faktor

Hier kann man ETX-Werte von OLSR manuell manipulieren, indem man den gemessenen Link-Quality-Wert um einen vorgegebenen Faktor für be-

stimmte Nachbarn kleiner rechnet. Dadurch werden die errechneten ETX-Werte größer (also schlechter). Damit kann man erreichen, dass OLSR seltener in eine bestimmte Richtung routet. Das wird vor allem dann angewendet, wenn Olsrd immer wieder zwischen zwei Internet-Gateways umschaltet und deshalb Verbindungen abbrechen. Es ist ein Notbehelf, der nicht wirklich schön ist. Wenn man zum Beispiel verhindern möchte, dass OLSR über die Knoten 104.0.0.1 und 104.0.0.2 routet, kann die Linkqualität von 104.0.0.1 und 104.0.0.2 durch einen Faktor von 0.1 bis 0.9 verkleinern:

Beispiel:

```
104.0.0.1:0.3;104.0.0.2:0.3
```

Man kann auch den umgekehrten Weg gehen: sämtliche Nachbarn pauschal schlechter rechnen und nur bestimmte Knoten durch einen höheren Faktor bevorzugen:

```
104.0.0.0/8:0.5; 104.0.0.5:1.0
```

Hysterese, Hysterese-Tempo, Oberer Grenzwert, Unterer Grenzwert

Diese Einstellungen sind deaktiviert, wenn ETX ausgewählt wurde. Da es sich um OLSR-Altlasten aus RFC 3626 handelt, lohnt es nicht, darauf näher einzugehen.

DynGW

Das dynamische Gateway-Plugin. Ist diese Funktion aktiviert, agiert der Router als Internet-Gateway für das Mesh, solange er erfolgreich überprüfen kann, dass er Internetzugang über seinen Internetport bekommt. Stellt Olsrd fest, dass der Internetzugang nicht mehr funktioniert, werden keine HNA-Nachrichten mehr verschickt, die einen Zugang zum Internet annoncieren.

Nameservice

Dieser Schalter aktiviert das Nameservice-Plugin von Olsrd. Damit kann dieser Rechner dem Mesh mitteilen, wie er heißt. So können OLSR-Knoten nicht nur durch ihre IP-Adresse, sondern auch durch ihre Namen angesprochen werden. Ein sehr empfehlenswertes Plugin.

Httpinfo

Dieses Plugin erzeugt eine HTML-Seite mit den Statusinformationen des Olsrd. Diese können öffentlich über die Statusseite dieses Routers eingese-

hen werden. Dieses Plugin sollte außer in Ausnahmefällen immer laufen, denn es erleichtert den Anwendern die Überprüfung der Funkstrecken und des Netzwerkzustands ungemein.

Mcast-Forward

Dieses Plugin ermöglicht den Transport von Multicast-Paketen über Olsrd und soll für den Transport von Multimediadaten (Audiostreams, Videostreams) eingesetzt werden. Bislang experimentell.

OLSR Traffic Shaping

Erlaubt den bevorzugten Transport von OLSR-Protokollnachrichten vor anderen Datenpaketen. Verbessert die Stabilität des Routingprotokolls. Sollte eingeschaltet sein.

Fischaugen-Routing

Reduziert den Netzwerk-Overhead und die Rechenlast von Olsrd (siehe Kapitel 3.2.5 auf Seite 42). Erhöht die Skalierbarkeit und die Stabilität des Routing. Vermeidet weitestgehend Routing-Loops. Sollte immer eingeschaltet bleiben.

Dijkstra-Optimierung

Reduziert die Rechenlast, indem die Topologiedatenbank seltener durchgerechnet wird – kann für Netzwerke mit weniger als 150-200 Knoten ausgeschaltet bleiben. Bei größeren Meshes sollte man sie unbedingt verwenden. Hat leider leicht negative Auswirkungen auf die Qualität und Reaktionsgeschwindigkeit des Routing.

6.3.6 Drahtlos

In dieser Sektion wird das Verhalten der WLAN-Karte eingestellt. Die ersten sieben Punkte sind schnell erklärt.

Im Pull-Down-Menü **WLAN-Protokoll** kann die IP-Adresskonfiguration der WLAN-Karte eingestellt werden. Für den Einsatz in einem Meshnetzwerk kommt nur die Einstellung **Statisch** für die Eingabe einer statischen IP-Adresse und der zugehörigen Netzmaske in den darunter liegenden Eingabefeldern in Frage.

Die **WLAN-IP-Adresse** ist die Adresse der WLAN-Karte. Diese erhält man durch die Koordination mit den anderen Teilnehmern des Meshnetzwerks,

beispielsweise 104.134.156.178. Dazu kommt die zur Netzwerkadresse des Netzwerks passende **WLAN-Netzmaske**, z. B. 255.0.0.0 für ein Netzwerk der Klasse A mit bis zu 16,7 Millionen Teilnehmern.

Die Einstellung **WLAN-Default-Route** darf nur verwendet werden, wenn dieses Gerät nicht als Meshrouter eingesetzt wird. Ansonsten führt das Setzen einer WLAN-Default-Route früher oder später zu Routing-Loops. Bei einem Router für ein Meshnetzwerk unbedingt leer lassen!

Die Auswahlmöglichkeit **WLAN-Modus** gibt es, da man den Router prinzipiell auch mit der Freifunk-Firmware als Accesspoint oder Accesspoint-Client einsetzen kann. Für einen Meshrouter ist die Einstellung des WLAN-Modus aber grundsätzlich **Ad-Hoc**.

Um eine reine Punkt-zu-Punkt-Verbindung zwischen zwei Knotenpunkten über eine Richtfunkstrecke herzustellen, kann ein Router als Accesspoint und die Gegenstelle als Accesspoint-Client arbeiten. Das kann vorteilhaft sein, wenn man eine WLAN-Karte hat, die nicht richtig im Ad-Hoc-Modus funktioniert. Die meisten Karten arbeiten immerhin stabil als Accesspoint-Client.

Die **ESSID** ist die Extended Service Set ID – der Netzwerkname der Funkzelle, der in Bakensignalen mit übertragen wird, z. B. **leipzig.freifunk.net** oder **olsr.freifunk.net**. Sie muss bei allen Knoten, die zu einer Funkzelle gehören sollen, exakt gleich sein. Bei einem – noch so kleinen – Schreibfehler klappt die Verbindung nicht.

BSSID

Mit dem Menüpunt **BSSID** ist die IBSSID (Independent Basic Service Set Identifier) gemeint – die Cell-ID der Funkzelle. Korrekt funktionierende WLAN-Karten im Ad-Hoc-Hoc-Modus handeln sie üblicherweise über Zeitstempel aus (siehe Abschnitt 5.4.1 auf Seite 112). Hinter diesem Eingabefeld verbirgt sich der dort beschriebene IBSS-ID-Hack. Die Cell-ID wird in diesem Eingabefeld festgelegt, um Cell-Splits zu verhindern. Die Freifunker verwenden üblicherweise die Cell-ID 02:CA:FF:EE:BA:BE. Die IBSSID muss in allen Knoten einer Funkzelle gleich sein, auch darf es niemals eine zweite Funkzelle mit anderer ESSID und gleicher Cell-ID innerhalb überlappender Reichweite geben.

Kanal und Kartentyp

Der **Kanal** ist in Deutschland, Österreich und der Schweiz frei wählbar von 1 bis 13. Die Meshbetreiber müssen sich auf einen Kanal einigen, wenn alle Knoten in der gleichen Funkwolke arbeiten sollen. Günstig ist Kanal 13, da er seltener benutzt wird als die anderen.

Der **Kartentyp** ist meist 802.11bg, in seltenen Fällen kann auch auf 802.11a umgeschaltet werden (bei Systemen mit Dualband-WLAN-Karten).

Angaben zu Antennen und Sendeverhalten

Bei Geräten mit zwei Antennen kann mit den Menüpunkten **Empangsantenne** und **Sendeantenne** eine der angebotenen Einstellung gewählt werden, wenn man an einem Antennenanschluss eine gewinnsteigernde Antenne hat. Es ist aber nicht immer klar, welche Antenne der Router als linke oder rechte Antenne ansieht. Deshalb sollte unbedingt getestet werden (z. B. durch Abnehmen einer Antenne und Überprüfung der Signalstärke), ob auch wirklich der richtige Anschluss eingeschaltet ist. Betreibt man einen WRT mit beiden mitgelieferten Stummelantennen oder hat nur eine Antenne, sollte man **Auto** wählen.

Die Freifunk-Firmware erlaubt über das Webinterface die Einstellung der **Sendeenergie** zwischen 1 und 84 Milliwatt. Bei Antennen mit starkem Gewinn muss die Sendeleistung reduziert werden, um die zugelassene maximale effektive Strahlungsleistung nicht zu überschreiten, siehe Abschnitt 7.6.1 zum Link-Budget im Kapitel 7.

Ein besonderes Bonbon der Freifunk-Firmware verbirgt sich hinter der Eingabemaske **Entfernung (Meter)**. Hier kann das Timing von 802.11 für Langstreckenverbindungen eingestellt werden, siehe Abschnitt 7.6.5. Theoretisch erlauben die vorgegebenen Zeitfenster im Übertragungsprotokoll von 802.11 nur Datenverbindungen bis zu einer Entfernung von 1,2 Kilometer, praktisch sind es bei den meisten Geräten auch ohne Modifkationen am Timing ein paar Kilometer. Über dieses Eingabefeld kann das Timing für Funkstrecken von beliebiger Länge angepasst werden. Diese Funktion ist nur für Langstreckenverbindungen interessant. Für den Betrieb in einem Mesh sollte dieses Feld leer bleiben.

Funkmodus und Übertragungsraten

Im Punkt **Funkmodus** kann gewählt werden, ob die WLAN-Karte nur mit anderen WLAN-Karten nach 802.11g funkt (**Nur G-Modus**) oder ob sie sich abwärtskompatibel zu 802.11b verhält (**B-Modus und G-Modus**). Auch der Betrieb nur mit den Datenraten von 802.11b ist möglich (**Nur B-Modus**). Bei manchen Geräten gibt es darüber hinaus auch einen proprietären Modus mit höheren Datenraten, als sie in den Standards 802.11b und 802.11g vorgesehen sind.

802.11g ist nur bei Funkstrecken mit hohem Signalpegel eine Überlegung wert und sonst eher hinderlich, da 802.11b bei schlechten Signalverhältnissen schneller arbeitet. Die bevorzugte Einstellung in einem Mesh ist der zu 802.11b abwärtskompatible g-Modus.

Manche Betreiber von Hotspots verwenden die Funktion **(E)SSID senden**, um den Namen ihrer Funkzelle vor unerwünschten Mitnutzern zu verstecken. Damit lassen sich ohnehin nur Gelegenheitsnassauer aussperren; in einem Mesh – zumal einem Community-Netzwerk – ist diese Funktion ganz sinnlos. Hier sollte immer der Name der Funkzelle gesendet werden. Außerdem sollte dieser so informativ wie möglich sein. Am besten verweist die ESSID direkt auf eine Adresse im WWW, wo interessierte WLAN-Nutzer weitere Informationen vorfinden.

Die **Basisrate** ist die Datenrate, mit der die Beacons (Bakensignale) und andere Broadcasts (beispielsweise die Protokollnachrichten von OLSR und B.A.T.M.A.N.) übertragen werden. Zu Koordination der Frequenznutzung werden Broadcasts mit der niedrigsten Datenrate gesendet, da sie die größte Reichweite und daher den besten Koordinationseffekt haben, der unter anderem zur Vermeidung von Kollisionen im Funkkanal dringend erforderlich ist. Im g-Modus ist die niedrigste Datenrate 6 Megabit, im b-Modus 1 Megabit. Die Einstellung **Rate je nach WLAN-Modus** ist empfehlenswert.

Bei der **Übertragungsrate** ist die Grundeinstellung **Automatisch**, damit fährt man in der Regel am besten – die WLAN-Karten passen sich dann den jeweils vorherrschenden Übertragungsbedingungen an. Hat eine WLAN-Station im Ad-Hoc-Modus mehrere Nachbarn, müssen diese mit unterschiedlichen Geschwindigkeiten angesprochen werden, je nach Qualität der jeweiligen Funkstrecke zum Nachbarn. Darum kümmert sich die Firmware beziehungsweise der Treiber der WLAN-Karte – sofern diese etwas taugen. In seltenen Fällen kann es sinnvoll sein, die Rate auf die niedrigste Einstellung zu setzen.

Steht der WLAN-Modus auf **B-Modus und G-Modus**, kann es vorkommen, dass eine Station ignoriert wird, die nur 802.11b beherrscht. Der **CTS-Schutz** verhindert das. Das kann aber den unerwünschten Effekt erzeugen, dass alle Stationen, die auch über den g-Modus verfügen, nun nur noch im b-Modus arbeiten.

Unter **Frame-Burst** kann man die Übertragungsgeschwindigkeit um einige Prozent erhöhen, indem mehrere Datenpakete zwischen zwei Stationen innerhalb kurzer Zeit in einem „Schwall" (Burst) aufeinanderfolgender Datenpakete verschickt werden. Normalerweise reservieren sich WLAN-Karten die Übertragungszeit auf dem Funkkanal für jedes einzelne zu übertragende Datenfragment getrennt. Ist Frame-Burst aktiviert, wird die Übertragungszeit von einer Station gleich für mehrere Datenfragmente reserviert.

In der Grundeinstellung ist diese Option deaktiviert. Sie ist für den Meshbetrieb nicht empfehlenswert, da dies potentiell zu einer ungerechten Verteilung der zur Verfügung stehende Bandbreite führt. Frame-Burst ist eine Funktion, die nicht in 802.11 standardisiert ist. Stationen, die Frame-Burst nicht verstehen, werden benachteiligt, da für sie das Übertragungsmedium die meiste Zeit belegt ist. Bei einem Langstreckenlink (Punkt-zu-Punkt-

Verbindung) kann diese Einstellung leichte Vorteile bei der Übertragungsgeschwindigkeit bringen, da die Datenübertragung zwischen den Stationen weniger Koordinationszeit benötigt.

Das **Beacon-Intervall** ist das Intervall, in dem Bakensignale zur Koordination und zum Announcement der Funkzelle verschickt werden. Die Grundeinstellung ist 100 Millisekunden.

WLAN-Karten können sich in einen Schlafzustand versetzen, um Energie zu sparen. Damit sie eingehende Nachrichten nicht „verschlafen", kann ihnen durch den Accesspoint mit dem **DTIM-Intervall** mitgeteilt werden, wann sie wieder mit der Übertragung von Informationen rechnen müssen. Bis zum nächsten Weckruf puffert der Accesspoint die anstehenden Datenübertragungen. Diese Einstellung ist nur relevant, wenn der Router als Accesspoint arbeitet.

Paketgrößen und Fragmentierung

Der Punkt **Frag.-Schwelle** steht für Fragmentierungs-Schwelle. Große Datenpakete haben eine höhere Wahrscheinlichkeit, bei der Übertragung über eine schwache Funkstrecke verloren zu gehen, als kleinere Datenpakete. Darum ist es unter schlechten Funkbedingungen (niedriges Signal-Rausch-verhältnis, häufige Störungen durch konkurrierende Funkanwendungen) besser, viele kleinere Datenhäppchen zu übertragen.

WLAN-Karten sind in der Lage, Datenpakete in mehrere kleinere Fragmente zu teilen und diese einzeln zu übertragen. Geht ein einzelnes Fragment verloren, muss nur dieses erneut übertragen werden. Die Fragmentierungsschwelle gibt an, ab welcher Größe eines Datenpakets dieses in einzelne Fragmente aufgeteilt wird. Eingestellt werden kann ein Wert zwischen 256 und 2346 Byte. Da ein IP-Paket meistens eine Größe von 1500 Byte hat, muss ein entsprechend kleinerer Wert gewählt werden, damit die Fragmentierung einsetzt.

RTS-Schwelle ist die Schwelle für die Paketgröße, ab der das Verfahren Request-to-Send/Clear-to-Send (RTS/CTS) beim Senden angewendet wird. Es funktioniert im Mesh nicht zufriedenstellend und sollte deaktiviert sein (siehe Abschnitt 2.1.3).

Der **MTU-Wert** (Maximum Transfer Unit) gibt die maximale Größe eines IP-Pakets vor. Die MTU-Größe ist von der Art der Netzwerkhardware abhängig. Bei Ethernet und WLAN beträgt die maximale Paketgröße 1500 Byte plus 18 Byte zur Komplettierung des Ethernet-Frame. Bei der MTU-Eingabe muss man die 18 zusätzlichen Byte des Ethernet-Frame jedoch nicht berücksichtigen. Über Multihop-Routen mit mehreren Hops kann es vorteilhaft sein, die MTU zu verkleinern (siehe auch den Abschnitt zur MTU auf Seite 25 im Abschnitt 2.1.2).

6.3.7 LAN

In dieser Sektion des Verwaltungsmenüs wird das Verhalten der Ethernet-Schnittstelle konfiguriert. Bei Asus WL-500 Premium, Buffalo WHR-G54S und dem Linksys WRT54GL ist die Ethernet-Schnittstelle als Switch mit vier nebeneinander liegenden Ethernetports ausgeführt. Diese Sektion ist nicht zuständig für die Konfiguration des abgesetzten einzelnen LAN-Ports, der mit „WAN" oder „Internet" gekennzeichnet ist.

Adressen, Netzmasken und Routen

Unter **LAN-Protokoll** kann man die Autokonfiguration der LAN-Schnittstelle per DHCP einstellen (**DHCP-Server abfragen**). Viele Anwender aktivieren fälschlicherweise diese Funktion. Gibt es keinen DHCP-Server im Netzwerk, ist der Router dann nicht mehr über die LAN-Schnittstelle erreichbar, da er keine IP-Adresse zugewiesen bekommt. Für die allermeisten Anwendungsfälle muss hier **Statisch** stehen.

Auf keinen Fall darf man den Router Asus WL500G Premium so konfigurieren, dass er seine LAN-Adresse über DHCP erhält. Dessen Bootloader verwendet die hier konfigurierte IP-Adresse. Ist die Einstellung DHCP, ist keine IP-Adresse gesetzt und der Bootloader nicht mehr über Ethernet ansprechbar.

LAN-IP ist das Eingabefenster für die manuelle Einstellung der LAN-IP. Bei Geräten mit 4-Port-Switch ist diese IP-Adresse die gemeinsame Adresse aller vier Ethernetports. Die meisten Anwender haben hier eine der klassischen – oder eher berüchtigten – Adressen aus einem Netzwerk der Klasse C stehen, also 192.168.0.1 oder 192.168.1.1. Wenn man den IP-Bereich am LAN-Port des Routers über das Mesh via HNA bekannt machen möchte, dürfen die gleichen Netzwerkadressen keinesfalls an anderer Stelle verwendet werden, da sonst IP-Kollisionen im Netzwerk bestehen. Es lohnt sich, hier etwas kreativer zu sein. Wie wäre es beispielsweise mit einem kleinen Subnetz aus dem Bereich 172.16-31.xxx.xxx? Die passende Netzmaske trägt man unter **LAN-Netzmaske** ein. Für die Berechnung von gültigen Netzwerkadressen und Netzmasken gibt es einfache Programme und Online-Kalkulatoren wie das Linux-Programm `ipcalc`.

Wer nicht ganz genau weiß, was er tut, lässt das nachfolgende Feld **LAN-Default-Route** leer. Für den Uplink via Ethernet zu einem DSL-Modem oder Ethernet-Segment mit Internetanschluss sollte stattdessen der WAN-Port (oder Internet-Port) verwendet werden.

Unter **Statische Routen** können Routingeinträge mit der Syntax *`Netzwerkadresse:Netzwerkmaske:Gateway-IP:Metrik:Interface`* hinzugefügt werden. Der Name für das Interface der vier Ethernetports ist `vlan1`, die Metrik ist normalerweise 0. Es können selbstverständlich auch Hostrou-

ten (zur IP-Adresse einer einzelnen Maschine) angegeben werden. Mehrere Routingeinträge werden durch Leerzeichen getrennt. Hier ein Beispiel für eine Hostroute zum Rechner 10.23.23.23 und eine Route zum Netzwerksegment 10.34.0.0:

```
10.23.23.23:255:255:255.255:172.16.134.100:0:vlan1
10.34.0.0:255:255.0.0:172.16.134.100:0:vlan1
```

NAT, Firewall und spezielle DHCP-Optionen

Die Network Address Translation (NAT) ist im Standardfall eingeschaltet und kann mit **NAT ausschalten** deaktiviert werden. Die meisten Anwender verwenden lokale Adressen auf dem LAN-Interface, die aus dem Mesh nicht erreichbar sind. Diese werden durch NAT durch die Firewall im Router maskiert. Der Zugriff aus dem Ethernetsegment hinter der Firewall dieses Routers auf das Mesh und das Internet sind möglich. Umgekehrt geht das aber nicht – bzw. nur, wenn eine Verbindung aus dem lokalen LAN auf einen Rechner auf der anderen Seite der Firewall initiiert wird. Dadurch sind die Rechner im LAN durch Zugriffe aus dem WLAN geschützt. Andererseits sind sie auch von außen nicht zu erreichen.

Die Firewall sperrt – wie es für eine Firewall üblich ist – Zugriffe von außen. Die meisten Anwender werden das zu schätzen wissen. NAT ist eine der Firewall-Funktionen. Wird über die Checkbox **Firewall ausschalten** die Firewall „ausgeschaltet", dann deaktiviert das zwar die Filterfunktion, aber NAT bleibt eingeschaltet. Werden auf der LAN-Seite Adressen aus dem gleichen IP-Bereich der WLAN-Seite, also aus dem Mesh verwendet, ist die Firewall nicht eingeschaltet.

DHCP-Start-IP ist die niedrigste IP-Adresse, die von dem DHCP-Server dieses Routers an die Rechner am Ethernetport vergeben wird.

Die **DHCP-Benutzeranzahl** ist die maximale Anzahl IP-Adressen, die vom DHCP-Server dieses Routers ausgegeben wird. Beträgt dieser Wert beispielsweise 50, so wird der Router bei einer Startadresse von 172.16.134.100 die Adressen 172.16.134.100 bis 172.16.134.149 an die lokalen Rechner verteilen.

Die Gültigkeitsdauer einer über DHCP herausgegebenen IP-Adresse in Sekunden stellt man unter **DHCP-Lease-Dauer** ein. Bleibt das Feld leer, wird ein Wert von 43 200 Sekunden (12 Stunden) verwendet. Mit dieser Einstellung dürften kaum Probleme auftreten; Ausnahmen könnten Situationen sein, in denen immer wieder neue Rechner nach einer Adresse fragen. In diesem Fall wäre eine kürzere Gültigkeitsdauer von Vorteil.

6.3.8 WAN

In diesem Menübereich wird die Konfiguration des WAN-Port (oder Internet-Port) vorgenommen, der üblicherweise den Uplink zum Internet oder einem größeren LAN-Segment herstellt. Die beiden häufigsten Einsatzszenarien sind der Anschluss an ein DSL-Modem oder an einen Hub oder Switch, der über DHCP Adressen herausgibt.

Für die Einwahl ins Internet über ein externes DSL-Modem muss zusätzliche Software über das Paketmanagement installiert werden, dieser Fall wird direkt im Anschluss an diesen Abschnitt behandelt.

Das Pull-Down-Menü **WAN-Protokoll** legt die Methode fest, mit der der Router die IP-Konfiguration für den WAN-Port erhält, beim Anschluss an ein LAN-Segment mit eingerichtetem DHCP-Server also **DHCP-Server abfragen**. Die Eingabefelder **WAN-IP**, **WAN-Netzmaske** und **WAN-Default-Route** verlangen im Falle einer manuellen WAN-Konfiguration (ohne DHCP) die entsprechenden Daten analog zu Abschnitt 6.3.7.

Die Bezeichnung des Eingabefelds **RJ45 Anschlüsse** ist etwas verwirrend, RJ45 ist schließich nur die Bezeichnung der Steckverbindernorm von Ethernetanschlüssen. Eine Bemerkung vorweg: Mit den Grundeinstellungen liegt man in den meisten Fällen richtig. Nur für Benutzer, die fortgeschrittene Netzwerkkonfigurationen einrichten möchten, dürften die folgenden Erklärungen interessant sein.

Virtual LAN

Unter Linux lassen sich alle fünf Ethernetbuchsen eines WRT, Buffalo etc. logisch trennen oder zusammenschalten (auch als brücken bzw. bridgen bezeichnet) – als wären es Anschlussbuchsen an unterschiedlichen Ethernetswitches, die zu getrennten Ethernetsegmenten gehören. Diese Technik wird als VLAN (Virtual LAN) bezeichnet und ist bei professionellen Ethernetswitches zu finden. Große LAN-Netzwerke wie beispielsweise in Universitäten oder Unternehmen machen oft von VLANs Gebrauch.

In den Grundeinstellungen legt die Freifunk-Firmware zwei VLAN-Schnittstellen `vlan0` und `vlan1` an. (Wer möchte, kann über die Kommandozeile auch noch weitere VLAN-Interfaces anlegen.) Der für den Anschluss einer Internetverbindung vorgesehene WAN-Port (oder Internet-Port) wird von der Freifunk-Firmware als einzige RJ45-Buchse zu VLAN-Interface 0 (`vlan0`) zugeordnet. Die Ethernetports 1 bis 4 sind dem VLAN-Interface `vlan1` zugeordnet, also gebrückt. Sie verhalten sich daher so, wie es von den meisten Anwendern erwartet wird: wie ein Switch mit vier Ports.

Es können aber auch weitere Ethernet-Ports aus dem VLAN-Segment 1 abgetrennt und zum virtuellen LAN-Segment `vlan0` hinzugefügt (gebrückt) werden. Sie verhalten sich dann mit dem Internet-Port zusammen wie Ports

an einem einfachen, kleinen Switch. Die Eintragung **0 5** ist die Grundeinstellung. Sie besagt, dass zum VLAN-Interface 0 (`vlan0`) nur der Anschluss 5 (der Internet-Anschluss) gehört. Eine Eintragung von **0 4 5** ordnet den Ethernet-Port 4 und den WAN-Port dem VLAN-Interface `vlan0` zu.

Zugriffsrechte

Die Checkbox **SSH erlauben** gestattet den Zugriff aus dem Internet oder WAN auf das Kommandozeileninterface des Routers per Secure Shell (SSH), **HTTP erlauben** ermöglicht den Zugriff aus dem Internet oder WAN auf das Webinterface des Routers, und bei aktiviertem **Ping erlauben** antwortet der Router auf ICMP-Anfragen (Ping) aus dem Internet.

6.3.9 Publizieren

Die Freifunk-Firmware bietet die Möglichkeit, selbst gestaltete Webseiten mit dem Webserver des Routers zu veröffentlichen. Das Webinterface des Routers liefert unter der Rubrik **Publizieren** eine Anleitung dafür.

6.3.10 Software1

Unter dieser Rubrik können zusätzliche Programme im Router installiert werden. Die Installationsdateien müssen auf dem lokalen Arbeitsplatzrechner gespeichert sein. Sie werden über den Webbrowser in den Router geladen und anschließend installiert. Diese Vorgehensweise ist nicht das bevorzugte Installationsverfahren, da eventuelle Abhängigkeiten der zu installierenden Softwarepakete auf diese Weise nicht automatisch erfüllt werden.

Viele Programme setzen voraus, das andere Programme oder Programmbibliotheken zusätzlich installiert sind. Sie funktionieren nicht wie erwartet, wenn die Abhängigkeiten nicht aufgelöst sind.

Der Vorteil dieses Menüs ist, dass der Router keinen Internetzugang zum Installieren von Softwarepaketen benötigt, da diese manuell aus dem Arbeitsplatzrechner hochgeladen werden.

Über die Schaltfläche **freifunk-reeommended** können vier zusätzliche empfohlene Softwarekomponenten nachinstalliert werden:

- DNS/DHCP-Server
- rrdtool für grafische Darstellung von Routerstatistiken
- OLSR-Viz, um OLSR-Topologiegrafiken über das Webinterface des Routers zu erzeugen

- ein Kommandozeilentool namens `horst` (Horsts OLSR Radio Scanning Tool), ein überaus nützliches WLAN-Scanprogramm ähnlich Netstumbler oder Kismet, aber für den Einsatz in Meshnetzwerken optimiert

Diese Installation erfolgt automatisch und nicht über lokal gespeicherte Installationsdateien. Die Installation von von `freifunk-recommended` setzt also einen Router mit funktionierendem Internetzugang voraus.

6.3.11 Software2

Dieses Menü bietet die Möglichkeit, weitere Softwarepakete per Mausklick zu installieren. Der Router lädt sich die benötigten Installationsdateien aus dem Internet und erfüllt selbständig eventuell bestehende Paketabhängigkeiten. Bevor neue Softwarepakete installiert werden, sollte die Liste der zur Verfügung stehenden Softwarepakete erneuert werden – über den Link **Liste aktualisieren** am unteren Ende der Webseite. In der Liste der installierbaren Programme tauchen nur Installationspakete auf, die im Softwareverzeichnis auf den Servern der Freifunk-Firmware oder des OpenWRT-Projekts enthalten sind.

6.3.12 Firmware

Unter dieser Rubrik findet man die Möglichkeit, per Webinterface ein neues Firmware-Image in den Router einzuspielen. Dies funktioniert auch bei den Routermodellen, bei denen die Erstinstallation der Freifunk-Firmware noch über TFTP erfolgen musste. Über die Schaltfläche **Durchsuchen** kann – wie bei der Installation über das Webinterface bei der Linksys-Firmware – ein neues Firmware-Image ausgewählt werden.

6.3.13 Neustart

Es gibt unterschiedliche Optionen für den Neustart. Ein **Einfacher Neustart** genügt, um die in den letzten Abschnitten beschriebenen Veränderungen zu übernehmen. Beim **Neustart im ReadOnly-Modus (für Firmware-Update)** wird der beschreibbare Teil des Flash-Dateisystems nach dem Neustart so in das Dateisystem eingebunden (gemountet), dass nur davon gelesen werden kann. Dies ist erforderlich für das Einspielen einer neuen Firmware über das Webinterface. **Neustart mit Initialisierung („Festplatte formatieren", Dauer: 2 Minuten)** löscht den Inhalt des Flashspeichers. Der Bootloader CFE und das NVRAM bleiben erhalten. Der **Neustart im Failsafe-Modus** wird in Abschnitt 6.7.1 beschrieben.

Die Option **Neustart mit den Grundeinstellungen des Bootladers** löscht den Inhalt des NVRAM und stellt die Fabrikeinstellungen wieder her. Danach

sind sämtliche vorgenommenen Einstellungen gelöscht. Diese Option funktioniert nur mit den Routern Linksys WRT54G/GL, WRT54GS und Siemens SE505. Der SE505 speichert die MAC-Adressen von WLAN-Karte und Switch in den NVRAM-Variablen `et0macaddr` und `il0macaddr`. Diese müssen also vor dem Löschen des NVRAM manuell gesichert und danach wiederhergestellt werden.

6.4 Gateway-Konfiguration

6.4.1 Internet-Gateway über DSL

Das häufigste Szenario für einen Internet-Gateway in einem Community-Mesh ist der Anschluss eines Routers an ein externes DSL-Modem. Dieses wird mit einem gewöhnlichen Ethernet-Patchkabel an den Internet-Port angeschlossen.

Die meisten ISPs verwenden für DSL das Protokoll PPPoE. Die Treibermodule und Programme für PPPoE sind nicht im Installations-Image der Freifunk-Firmware enthalten, um im Flash-Speicher Platz zu sparen. Die benötigte Funktionalität kann über das Paketmanagementsystem nachinstalliert werden.

Nach der Installation des PPPoE-Pakets taucht in der Rubrik **WAN** im Auswahlmenü **WAN-Protokoll** zusätzlich die Option **PPPoE** auf. Wird sie ausgewählt, erscheinen nach der Installation neue Eingabefenster für die Zugangsdaten des Providers und das Kennwort.

Unter der Rubrik **OLSR** muss nun noch die Option **DynGW** eingeschaltet werden. Außerdem müssen im Menü **LAN** die Schaltflächen **NAT** und **Firewall** aktiviert sein. Damit ist nach dem Neustart der Konfigurationsprozess für ein einfaches Gateway über ein DSL-Modem bereits abgeschlossen.

6.4.2 Internet-Gateway über ein bereits existierendes LAN

Der Internet-Port des Routers wird mit dem existierenden LAN über ein Patchkabel verbunden. Ist im äußeren LAN-Segment ein DHCP-Server vorhanden, muss lediglich im Menü WAN-Protokoll die Option **DHCP-Server abfragen** ausgewählt werden. Die Option **DynGW** für OLSR muss eingeschaltet werden, damit der Router dem Mesh den Internetzugang über HNA annonciert. Im Normalfall werden auch die Optionen **NAT** und **Firewall** unter **Verwaltung | LAN** eingeschaltet.

Eine statische WAN-Konfiguration unterscheidet sich von dieser Vorgehensweise nur dadurch, dass in der Rubrik **WAN** die IP-Konfiguration manuell eingegeben werden muss.

6.4.3 Fortgeschrittene Konfiguration – das Freifunk-Gateway-Plugin

Die bisher beschriebenen einfachen Wege zur Einrichtung eines Gateways bieten nur minimale Flexibilität. Nach einer Weile stellt sich bei den Gatewaybetreibern üblicherweise der Wunsch nach Optimierungen ein.

Die Funktionalität und Performance eines Gateways kann durch die Installation weiterer Softwarepakete ausgebaut werden. Vor allem Traffic-Shaping ist interessant, um die Verteilung der Bandbreite unter den Gatewaynutzern besser und fairer zu regeln. Viele Haushalte haben eine DSL-Flatrate, die sie zum großen Teil gar nicht auslasten können. Sie sind gerne bereit, die zur Verfügung stehende Bandbreite mit einem Community-Netzwerk zu teilen, wollen dabei aber nicht selbst benachteiligt werden.

Für diesen Fall gibt es das Softwarepaket `freifunk-gateway-de`, das viele nützliche Funktionen bereitstellt. Insgesamt zehn zusätzliche Softwarepakete müssen mitinstalliert werden, um sämtliche Abhängigkeiten zu erfüllen, weshalb die Installation am besten über die Rubrik **Software2** im Verwaltungsmenü von der Hand geht. Die Paketabhängigkeiten werden so automatisch erfüllt. Die Software erweitert das Verwaltungsmenü um die Rubrik **Gateway**, in der zusätzliche Eingabe- und Auswahlfelder stehen:

Hinter **ext. Interface** verbirgt sich, wenig überraschend, der WAN-Port, über den der Router mit dem Internet verbunden ist, üblicherweise `vlan1`.

In **URL-Whitelist** wird der Pfad oder die URL zu einer einfachen Textdatei eingetragen, die alle IP-Adressen auflistet, die den Internetzugang über dieses Gateway benutzen dürfen. Für alle anderen IP-Adressen ist das Gateway gesperrt. Die Datei muss das Format einer einfachen Liste haben – eine IP-Adresse pro Zeile. Ist das Feld leer, darf jede IP-Adresse auf das Internet zugreifen.

In das Eingabefeld **MAC-Blacklist** können die MAC-Adressen von WLAN-Karten direkter Nachbarn eingetragen werden, denen der Zugang zum Gateway verweigert werden soll. Das Sperren von MAC-Adressen ist nur bei direkten Nachbarn möglich, da MAC-Adressen nur bei direkten Funklinks sichtbar sind. **IP-Blacklist** hat dieselbe Funktion, nur dass hier IP-Adressen gesperrt werden können.

Die White- und Blacklistfunktionen sollten in einem Community-Netzwerk mit Weitsicht eingesetzt werden. Man riskiert sonst einen unangenehmen Kleinkrieg mit fortgeschrittenen Nutzern in der Nachbarschaft, denn IP- und MAC-Adressen können gefälscht werden. Sollte der ausgegrenzte „Störer" anfangen, registrierte IP-Adressen oder MAC-Adressen von anderen zu benutzen (zu „spoofen"), greifen die hier gebotenen Maßnahmen nicht, und das Netzwerk kann empfindlich gestört werden.

Ein offenes und freizügiges Netzwerk hingegen regt kaum zur Sabotage an. Störungen entstehen weitaus häufiger durch Unwissenheit als durch Bös-

willigkeit. Derartige Probleme werden besser auf der sozialen, zwischenmenschlichen Ebene („Layer 8" im Freifunk-Jargon) als durch technische Maßnahmen gelöst.

P2P-Traffic blocken

Der Punkt **P2P-Traffic blocken** hat die Aufgabe, Filesharing-Protokolle (Peer-to-Peer-Protokolle, P2P) wie E-Mule, Kazaa oder Bittorrent zu sperren. Diese können den Gateway für andere Internetnutzer durch Überlastung weitestgehend unbrauchbar machen. Das Problem tritt dann auf, wenn Peer-to-Peer-Programme so konfiguriert sind, dass sie unbegrenzt viele simultane Verbindungen zu anderen Peers aufbauen und sämtliche verfügbare Internetbandbreite aufbrauchen. Viele Anwender von P2P wissen nicht, dass man die Transferraten und die Anzahl der simultanen Verbindungen beschränken kann.

Die P2P-Filterfunktion erzeugt aber auf der CPU des Gateways eine hohe Rechenlast, da sie den Inhalt der Datenpakete analysieren muss, um zu überprüfen, ob P2P-Protokollinformationen enthalten sind. Die Sperre kann nur bekannte Protokolle blockieren, und diese auch nur dann, wenn der Inhalt der IP-Pakete nicht komprimiert und/oder verschlüsselt ist[5].

Prinzipiell ist die Kontrolle von P2P-Traffic ein Katz-und-Maus-Spiel, bei dem die Programmierer der P2P-Protokolle durch immer neue Raffinessen gewinnen. Filesharing ist nicht per se illegal oder böse. Viele Linux-Distributionen setzen inzwischen auf das P2P-Protokoll Bittorrent, um CD-Images oder DVD-Images zum Download anbieten zu können. Bittorrent ist ein hochinteressantes Verfahren, um Anwendern große Datenmengen preiswert zur Verfügung zu stellen.

Traffic-Kontrolle und Traffic-Shaping

Die **Accounting**-Funktion zeichnet auf, wieviel Datenverkehr von jeder IP-Adresse im Mesh über dieses Gateway ins Internet gesendet oder von dort empfangen wurde. Die gewonnenen Daten werden jedoch nicht permanent gespeichert und sind bei einem Router-Neustart verloren. Sie können aber manuell gesichert werden.

Das Accounting sollte nur den Datenverkehr messen, der über das WLAN-Interface zum Internet geroutet wird. In den Feldern **Filter(eingehend)** und **Filter(ausgehend)** werden jene IP-Bereiche eingetragen, die nicht vom Accounting erfasst werden sollen. Die privaten LAN-Adressbereiche wie 192.168.0.0 tauchen hier schon auf. Daran muss also nichts geändert werden, außer es kommt ein noch nicht angegebener Adressbereich zum Einsatz.

[5] Details zu dem verwendeten Verfahren sind unter der URL `http://ipp2p.org/` zu finden.

Verfügbare Internetbandbreite lässt sich zwischen dem Mesh und den lokalen Benutzern aufteilen. Dazu gibt man im Pull-Down-Menü **Bandbreite** die Gesamtbandbreite des Internetanschlusses an und legt unter **Freigeben** den prozentualen Anteil fest, der dem Mesh zur Verfügung stehen soll.

6.5 Konfiguration auf der Kommandozeile

Um die meisten der im Anschluss beschriebenen Einstellungen vornehmen zu können, muss man sich über SSH mit dem Kommandozeilen-Interface des Routers verbinden. Auch wenn das Webinterface der Freifunk-Firmware viele Einstellmöglichkeiten hat, macht es lediglich eine Untermenge der Konfigurationsmöglichkeiten zugänglich. Nur auf der Kommandozeile ist es beispielsweise möglich, mit einem Texteditor selbst Hand an die Konfigurationsdateien des Systems zu legen, selbst Skripte zu erstellen etc. Als Editor steht nur `vi` in der Freifunk-Firmware zur Verfügung, den Neulinge ohne Anleitung[6] kaum bändigen können.

Um eine SSH-Verbindung mit dem Router herzustellen, steht unter Windows das Freeware-Programm PuTTY zur Verfügung. Unter Mac OS X und Linux ist das passende Programm `ssh` bereits installiert. Letzteres wird auf der Kommandozeile ausgeführt. In einem Textterminal wird der Befehl

```
ssh root@router-ip
```

eingegeben. Der Benutzername ist immer `root`, ein normales Benutzerkonto gibt es nicht. Die Verbindung per SSH wird verschlüsselt und kann – im Gegensatz zu Telnet – als abhörsicher bezeichnet werden.

6.5.1 Softwareinstallation auf der Kommandozeile

OpenWRT – und damit auch die Freifunk-Firmware – verwenden das Paketmanagementsystem iPKG mit dem Installationsbefehl `ipkg`. Die Befehlssyntax ähnelt `apt-get` unter Debian GNU/Linux oder Ubuntu. iPKG kann externe Softwarepakete, die nicht in der Paketliste der Freifunk-Firmware enthalten sind, über die Angabe einer Linkadresse installieren. Hier ein Beispiel, wie man den B.A.T.M.A.N.-Daemon[7] und das passende Webinterface installiert[8].

```
ipkg install http://freifunk.schmudde.com/ipkg/freifunk-batman_0.84-
beta.02.ipk
```

[6] `http://www.linux-root.de/elug/vi.html`
[7] `http://downloads.open-mesh.net/batman/stable/`
[8] `http://freifunk.schmudde.com/`

```
ipkg install libpthread
ipkg install  http://downloads.open-mesh.net/batman/stable/wrt-freifunk/
batmand_0.2-current.ipk
```

Wenn Softwarepakete in der Paketliste der Freifunk-Firmware vorhanden sind (wie hier `libpthread`), genügt es, den Paketnamen anzugeben.

6.5.2 Zugang vom Mesh ins LAN

In den Standardeinstellungen können zwar Computer am LAN-Port des Freifunk-Routers auf das Mesh zugreifen, aber nicht umgekehrt. Die Ursache ist neben der Firewall, die Rechner am LAN-Port vor Zugriffen aus dem Mesh beschützen soll, die Tatsache, dass zwischen LAN und WLAN NAT (*Network Address Translation*) eingesetzt wird.

NAT behindert den Zugriff aus dem Mesh

NAT ist bei privaten Internetzugängen notwendig, weil die Internet-Serviceprovider den privaten Endnutzern normalerweise nur eine öffentliche IP-Adresse pro Anschluss zukommen lassen, die im Internet geroutet wird. Daher wird man auf NAT zum WAN-Port des Routers hin selten verzichten können. Außerdem sind die Gefahren einer Attacke größer, wenn ein Arbeitsplatzrechner direkt ans Internet angeschlossen ist.

An privaten IP-Adressen, die in einem Mesh normalerweise verwendet werden, herrscht hingegen kein Mangel. Man kann daher auf NAT innerhalb eines Community-Netzwerks gänzlich verzichten – ausgenommen an den Gateways zum Internet hin. Das ist natürlich ein Problem für Anwender mit inhärent unsicheren Betriebssystemen oder besorgte Zeitgenossen, die den anderen Mesh-Nutzern nicht trauen.

Das Sicherheitsproblem liegt jedoch eher bei den angeschlossenen PCs als am Netzwerk. NAT bringt zwar eine gewisse Sicherheit (die ein geschulter Angreifer leicht aushebeln kann), wirkt aber auch störend, weil Rechner am LAN-Port des Routers nicht von außen erreichbar sind. Sollen einzelne Ports eines Rechner hinter einem NAT von außen zugänglich sein, muss aufwendiges Port-Forwarding eingerichtet werden. Dies ist dann nötig, wenn ein Server am LAN-Port des Routers dem Mesh seine Dienste anbieten soll.

Hier ein kleines, fast intuitiv verständliches Beispiel für Port-Forwarding ohne Erklärung der Optionen:[9] Als WLAN-Adresse des Meshrouters kommt

[9] Ausführliche Informationen zu Port Forwarding mit `iptables` findet man zum Beispiel in der deutschen Übersetzung von Rusty Russels NAT-Howto (`http://www.netfilter.org/documentation/HOWTO/de/NAT-HOWTO.html`).

10.140.131.4 zum Einsatz, der interne Rechner, der am LAN-Port des Routers hängt, trägt die LAN-Adresse 172.16.3.1. Ziel ist es, den TCP-Port 6000 des Rechners 172.16.3.1 von außen (aus dem Mesh) trotz NAT erreichbar zu machen:

```
iptables -t nat -A PREROUTING -d 10.140.131.4 -p tcp --dport 6000
-j DNAT --to 172.16.3.1:6000
```

Zusätzlich muss die Firewall geöffnet werden:

```
iptables -I FORWARD -d 10.140.131.4 -p tcp --dport 6000 -j ACCEPT
iptables -I FORWARD -d 172.16.3.1 -p tcp --dport 6000 -j ACCEPT
```

Diese Zeilen kann man in die Datei `/etc/local.fw` einfügen, damit die Veränderungen nach jedem Neustart wirksam werden. Das beschriebene Vorgehen führt lediglich dazu, dass ein Port aus dem LAN im Mesh erreichbar ist. Für das Mesh sieht es so aus, als würde der Router selbst den Dienst auf Port 6000 anbieten.

Eine Alternative zum Port-Forwarding stellt die Funktion OLSR-DMZ dar, die einzelne lokale Rechner aus dem Mesh erreichbar macht. Dann muss man keine Firewall-Regeln für einzelne Ports konfigurieren – mehr dazu im Abschnitt 6.3.5 ab Seite 139).

NAT ausschalten

Viele Anwendungen wie Voice over IP werden durch NAT behindert. Warum nicht bidirektionale Kommunikation zwischen den Netzwerkteilnehmern in einem privaten Netzwerk erlauben? Lässt man auf einem PC mit WLAN-Karte eines der Meshrouting-Protokolle laufen, ist man ohnehin direkt aus dem Mesh ansprechbar und (theoretisch) angreifbar. Man muss nicht zwingend NAT zwischen dem LAN- und dem WLAN-Anschluss des Routers verwenden.

Wird auf NAT in der Routerkonfiguration verzichtet, muss man sich um das Routing der LAN-Adressen in das Mesh kümmern. Es genügt in diesem Fall, die am LAN-Port oder über WLAN verbundenen lokalen Rechner per HNA-Announcement unter **Verwaltung | OLSR | HNA4** zu annoncieren (siehe Abschnitt 6.3.5, Seite 140).

Die letzte Hürde für die Kommunikation zwischen den Rechnern am LAN-Port und dem Mesh ist die Firewall, die Zugriffe vom WLAN auf das LAN blockiert. Um das Problem zu lösen, gibt es drei Möglichkeiten: Entweder schaltet man die Firewall ganz aus, man verwendet IP-Adressen aus dem Mesh (WLAN-Seite) auch auf der LAN-Seite oder man bearbeitet die Firewallregeln. Für den zweiten Weg, um also IP-Adressen aus dem Mesh auf der LAN-Seite zu verwenden, richtet man ein kleines Subnetz am LAN-Port

ein, das im großen Bereich des Mesh liegt. Angenommen die Netzwerkadresse des Mesh ist 10.0.0.0/8. Ein kleines Subnetz in diesem Bereich ist z.B. 10.1.3.100/30 – dieses Subnetz wird per OLSR-HNA announciert. Der Router bekommt die LAN-IP 10.1.3.101 und darf per DHCP die Adressen 10.1.3.102 und 10.1.3.103 herausgeben.

6.6 OpenWRT

OpenWRT kann man als eine Meta-Distribution für Wireless Router betrachten – ein vielseitiger Baukasten, auf den fortgeschrittene Linux-Bastler zurückgreifen können, um eigene Firmware-Images zu erzeugen.

Die Distribution verdankt ihren Namen dem WRT54G. Doch die OpenWRT-Entwickler sind nicht recht glücklich darüber, ausschließlich mit Linksys-Produkten assoziiert zu werden, und verweisen zu Recht darauf, dass ihr Projekt mittlerweile eine große Anzahl Modelle verschiedener Hersteller unterstützt. Sie möchten, dass die Anwender OpenWRT als Abkürzung für *Open Wireless Router Technology* verstehen.

Ausgangspunkt der Entwicklung von OpenWRT und anderen alternativen Firmware-Images war die Veröffentlichung der Quelltexte der originalen Linux-Firmware von Linksys. Linksys hatte es ursprünglich versäumt, seine Kunden darauf aufmerksam zu machen, dass die ersten WRT54G auf GNU/Linux und freier Software basierten, und die Quelltexte der Linux-Firmware nicht veröffentlicht, und damit gegen die GNU General Public License (GPL) verstoßen.

Als die amerikanische Free Software Foundation (FSF) Linksys dazu aufforderte, nicht länger gegen die GPL zu verstoßen, lenkte der Hersteller ein und veröffentlichte die Quelltexte. In der Folge entstanden viele alternative Firmware-Images (Sveasoft, FreeWRT, DDWRT) auf der Basis der Linksys-Sourcen, die oft nur geringfügig von der Originalfirmware abweichen. Damit hat die FSF offensichtlich die Firma Linksys zu ihrem Glück gezwungen. Populärer dürfte mittlerweile kein anderer Wireless Router sein. Ein billiges, aber ausgereiftes Consumer-Gerät wurde zu einer vielseitigen Hardwareplattform für hochinteressante drahtlose Netzwerklösungen – und zu einer Konkurrenz zu weitaus teureren Produkten aus dem WISP-Bereich (Wireless Internet Service Provider).

In der Originalfirmware des WRT54G und anderer Routermodelle enthält der Flashspeicher ein komprimiertes Dateisystem vom Typ cramfs (*Compressed RAM Filesystem*), welches nur gelesen werden kann. Beim Start wird sein Inhalt ins RAM des Routers entpackt. Von Anwendern auf den Router kopierte Programme sind nach dem nächsten Neustart verschwunden, da sie im RAM abgelegt werden. Anwender können das System nicht verändern, ohne ein neues Firmware-Image zu installieren; lediglich die Konfi-

gurationseinstellungen können in das 64 kByte große NVRAM geschrieben werden.

Die meisten alternativen Firmware-Images fügen dem cramfs-Dateisystem lediglich die eine oder andere Software hinzu und lassen manches aus der Originalfirmware aus Platzgründen weg. Darüber hinaus verändern die Programmierer das Design des Webinterface nach ihrem eigenen Geschmack („Branding") und erweitern es um Menüs, die zur Steuerung der hinzugefügten Software erforderlich sind.

OpenWRT geht dagegen sehr viel weiter und legt nur ein Grundsystem in ausschließlich lesbarer Form im Flash ab. Daneben lässt es im Flashspeicher Platz für ein Dateisystem, das wie auf einem PC Lese- und Schreibzugriffe gestattet. In diesem Bereich können Anwender eigenen Wünschen entsprechend Softwarepakete installieren. OpenWRT bietet das bereits erwähnte Paketmanagementsystem iPKG an, das auch OpenZaurus, Familiar Linux und OpenEmbedded verwenden. Dabei handelt es sich im Prinzip eine abgespeckte Variante des Debian-Paketmanagementsystems.

Die ältere OpenWRT-Version „White Russian" unterstützt ausschließlich Geräte mit Broadcom-Chipsätzen. Da die Freifunk-Firmware auf „White Russian" basiert, sind nur einige Router mit Broadcom-Hardware dafür geeignet. Der Broadcom-Chipsatz ist aber nicht nur in Routern anzutreffen, sondern z. B. auch in Webcams oder Festplatten mit WLAN-Anschluss. Wer sich mit OpenWRT beschäftigt und gerne bastelt, kann auch diese Geräte für den Einsatz in einem Mesh vorbereiten.

Mittlerweile unterstützt OpenWRT einen großen Teil der auf dem Markt befindlichen und höchst unterschiedlichen Routermodelle. Die neuere OpenWRT-Version „Kamikaze" arbeitet auch mit Chipsätzen anderer Hersteller, allerdings können abhängig vom Chipsatz Probleme mit dem Ad-Hoc-Modus auftreten. Wer neue Hardware anschaffen will, sollte auf der Webseite des Projekts einen Blick in die Tabelle mit den unterstützten Modellen werfen und ihren Status überprüfen.[10] Dort sind zwar sehr viele Routermodelle aufgelistet, oft aber mit dem Verweis `WiP` (*Work in Progress*). Wer durch Tests und Fehlerberichte aktiv an der Entwicklung einer Plattform mitarbeiten will, ist sicherlich willkommen.

Geräte mit Chips von Texas Instruments eignen sich wegen der hinlänglich beschriebenen Problematik mit dem Ad-Hoc-Modus (siehe Seite 111) nicht für ein Mesh. Bei Atheros-Systemen sieht es sehr viel besser aus, wenn es auch immer noch etwas an der Stabilität der Treiber für die drahtlose Schnittstelle mangelt. Doch gerade hier sind die OpenWRT-Entwickler – vor allem Felix Fietkau – recht umtriebig dabei, die Fehler des Madwifi-Treibers zu beseitigen.

Die Vorgehensweise beim Installieren eines Firmware-Images unterscheidet sich bei den meisten Geräten nicht grundlegend von der beim Flashen

[10] `http://wiki.openwrt.org/TableOfHardware`

der Freifunk-Firmware. Da es den Rahmen des Buches sprengen würde, auf die Einzelheiten jedes einzelnen Routermodells einzugehen, sei an dieser Stelle die Dokumentation im Wiki von `http://openwrt.org` empfohlen. Über die Liste der unterstützten Modelle findet man detailliertere Informationen zu den einzelnen Gerätetypen. Üblicherweise muss die Installation per TFTP erfolgen.

Ebenso wie bei der Freifunk-Firmware benötigen die verschiedenen Routermodelle auch verschiedene Firmware-Images. Bei OpenWRT gibt es diese jeweils in zwei Varianten, die sich in Größe und Funktionsumfang unterscheiden. Wer Platz für viele eigene Programme haben möchte, greift zur Minimalvariante.

Das Installieren eines Routingprotokolls wie B.A.T.M.A.N. oder OLSR gestaltet sich dank des iPKG-Paketmanagementsystems einfach, aber bis zu einem ausgereiften Meshsystem mit dem ganzen Komfort der Freifunk-Firmware ist es bei beiden Varianten noch ein Stück Arbeit.

Aktuelle B.A.T.M.A.N.-Versionen kann man von der Website herunterladen[11] oder direkt per iPKG installieren. Bei vielen Konfigurationstricks für den Einsatz in einem Mesh spart man sich Zeit, wenn man von der Freifunk-Firmware „abkupfert" – beispielsweise bei der Konfiguration von Olsrd und den Firewallregeln. Viele Installationspakete der Freifunk-Firmware lassen sich mit etwas Anpassungsarbeit auch unter OpenWRT „White Russian" einsetzen. Bei „Kamikaze" dürfte das mit dynamisch gelinkten Binärdateien aus der Freifunk-Firmware allerdings nicht glücken, da die verwendeten Versionen der Systembibliothek uClibc nicht zueinander binärkompatibel sind.

Anders als bei der Freifunk-Firmware müssen unter OpenWRT viele Konfigurationsschritte und Administrationsarbeiten auf der Kommandozeile per SSH ausgeführt werden. Verglichen mit dem ausgefeilten Webinterface der Freifunk-Firmware fällt die optionale Weboberfläche von OpenWRT spartanisch aus. Es ist also eher etwas für engagierte Linux-Poweruser.

6.7 Wenn etwas schiefgeht - Probleme mit der Router-Firmware

6.7.1 Router „verkonfiguriert" oder Passwort vergessen

Anwender sperren sich häufig selbst aus ihrem Router aus - etwa wenn sie nach längerer Betriebszeit das Administratorpasswort vergessen haben oder wenn fehlerhafte Netzwerkeinstellungen keinen Zugriff mehr auf das Gerät erlauben. In diesem Fall kann man bei der Originalfirmware von

[11] `http://downloads.open-mesh.net/`

Linksys durch längeres Drücken der Reset-Taste (10 bis 15 Sekunden) nach dem Einschalten die Fabrikeinstellungen wieder herstellen. Die im NVRAM gespeicherten Einstellungen werden also gelöscht und von den Fabrikeinstellungen überschrieben, die im Bootloader gespeichert sind. Danach lässt sich das Gerät wieder unter der IP-Adresse 192.168.1.1 mit dem Standard-Passwort `admin` ansprechen. OpenWRT und die Freifunk-Firmware haben diese Funktion nicht, stattdessen aber die wesentlich elegantere Bootoption `Failsafe`.

Der WRT hat eine „DMZ"-LED an der Gehäusefront, die einige Sekunden nach dem Einschalten zu leuchten beginnt. Wenn man in diesem Moment die Reset-Taste für ein paar Sekunden drückt, gelangt man in den Failsafe-Modus. Nun startet der Router mit der IP-Adresse 192.168.1.1, und man kann sich ohne Angabe von Benutzername und Passwort per Webinterface oder per Telnet verbinden, um das Konfigurationsproblem zu beheben oder das Passwort zu ändern.

Bei Geräten ohne DMZ-LED kann der richtige Moment zum Drücken der Reset-Taste anhand eines Broadcast-Pakets identifiziert werden, das der Router an die Zieladresse 192.168.1.255 Port 4919 über die Ethernetports verschickt. Dazu benötigt man ein Programm wie `tcpdump`, mit dem man den Netzwerktraffic auf der Ethernetschnittstelle beobachten kann. Übergibt man `tcpdump` die Option `-A`, wird der Inhalt der Pakete in ASCII-Text angezeigt:

```
tcpdump -i ethernetinterface_des_pc -A
```

Ist der Moment zum Drücken der Reset-Taste gekommen, erscheint in der `tcpdump`-Ausgabe ein Paket mit dem Inhalt `Press reset now, to enter failsafe!` Hat man die Reset-Taste rechtzeitig und ausreichend lange gedrückt, quittiert der Router den Vorgang mit der Meldung `Entering Failsafe!`

Der Bootloader CFE

Wenn der Router überhaupt nicht mehr funktioniert und außer dem schnellen Blinken einer einzelnen LED „kein Leben mehr in der Bude ist", helfen ein paar grundsätzliche Kenntnisse über den Startvorgang weiter.

Statt einer Festplatte besitzen die Router einen Flashchip, wie sie auch in USB-Memory-Sticks verwendet werden. Beim Einschalten wird der in den ersten Blöcken des Flash gespeicherte Bootloader CFE aktiv. Er erfüllt die gleiche Funktion wie das BIOS in einem PC.

Wenn der Bootloader gestartet ist, überprüft er den Inhalt des NVRAM (*Non-Volatile Random Access Memory*) anhand einer Prüfsumme. Dieser 64 kByte große Block am Ende des Flashspeichers enthält die Konfiguration des Systems. Nach der erfolgreichen Auswertung des NVRAM wird der

Linuxkernel ebenfalls anhand einer Prüfsumme kontrolliert. Ist sie korrekt, wird der Kernel ins RAM des Routers geladen und das Dateisystem im Flash gemountet. Anschließend werden die Init-Skripte ausgeführt.

Um ein neues Betriebssystem im Router installieren zu können, enthält der Bootloader einen TFTP-Server. Empfängt der CFE eine Firmware-Datei per TFTP-Upload, wird sie automatisch im Flash installiert. Der TFTP-Server des CFE ist beim WRT jedoch ab Werk ausgeschaltet. Es lässt sich beim Start des Routers also nicht einfach ein neues, funktionierendes System per TFTP flashen wie bei anderen Routern. Das wird zum Problem, wenn das im Flash installierte Firmware-Image aus irgendeinem Grund nicht mehr funktioniert – etwa weil beim Einspielen der Firmware per Webinterface ein Fehler aufgetreten ist. Wenn der Installationsprozess fehlschlägt, haben oftmals nur noch Bastler mit Lötambitionen die Möglichkeit, das hilflos blinkende Gerät wieder in einen benutzbaren Zustand zu versetzen.

Hat man die Freifunk-Firmware bereits einmal erfolgreich installiert, ist man beim weiteren Experimentieren schon eher auf der sicheren Seite. Durch das Setzen der Variable `boot_wait=on` im NVRAM kann auch bei Linksys-Geräten der TFTP-Server des Bootloaders aktiviert werden. Enthält das NVRAM diese Variable, bleibt der CFE nach der Auswertung des NVRAM drei Sekunden im TFTP-Modus stehen und wartet auf den Upload eines Firmware-Images. Nach dem Flashprozess schreibt die Freifunk-Firmware diesen Eintrag selbständig ins NVRAM. Danach kann die Firmware auch beim WRT54G/GL per TFTP neu installiert werden, wenn das System wegen eines Firmwareproblems nicht mehr startet.

Schlägt hingegen der erste Installationsversuch der Freifunk-Firmware fehl, befindet sich das System in einer Sackgasse – das System bootet nicht mehr, und der TFTP-Server ist noch nicht eingeschaltet. Aber keine Sorge: Auch im Fall eines völlig ruinierten Flashinhalts kann man den Router durch ein Programmierkabel wieder in einen brauchbaren Zustand versetzen, selbst wenn der Bootloader beschädigt ist.

6.7.2 Reparatur „gebrickter" WRTs

Symptomatisch für eine beschädigte Firmware ist eine dauerhaft schnell blinkende Power-LED ohne permanent leuchtende Kontroll-LEDs an den Ethernetports des WRT. In diesem Fall ist der Bootloader noch intakt, findet aber kein System vor, das er booten könnte, und man kommt nicht mehr an den TFTP-Server oder das Webinterface heran, um eine funktionierende Firmware zu installieren. Außer der fehlgeschlagenen Firmware-Installation bei abgeschaltetem `boot_wait` können auch falsche Einstellungen im NVRAM einen Start verhindern, z. B. wenn der Router durch geänderte NVRAM-Einstellungen übertaktet wurde.

Leuchten zusätzlich zur schnell blinkenden Power-LED sämtliche Kontroll-LEDs an den Ethernetbuchsen ohne Unterbrechung, ist auch der Bootloader defekt oder gelöscht. In jedem dieser drei Fälle lässt sich der vor sich hin blinkende Router nicht mehr von außen ansprechen.

In einer WLAN-Community ist es deshalb schön, mindestens eine Person zu haben, die in der Lage ist, mittels Lötkolben „zerflashte" Router wiederzubeleben. Im Internet und in gewissen Computerzeitschriften kursieren Anleitungen, wie man durch Kurzschließen der Anschlussbeinchen eines Flashchips den Router WRT54G/GL wieder zum Laufen bringen soll. Von dem Einsatz derartiger Holzhammermethoden sollte man absehen. Das Risiko, die bislang unbeschädigte Hardware beim Kurzschließen endgültig in Elektronikschrott zu verwandeln, ist groß. Bei einem beschädigten Bootloader ist dieser Trick ohnehin aussichtslos.

Mit ein klein wenig Lötarbeit kann man aber einen Router mit Firmware-defekt immer wieder neu beschreiben, egal wie verfranst die Routersoftware ist. Auch ein Gerät mit einem defekten Bootloader lässt sich wieder herstellen – jedenfalls solange das Gerät noch nicht durch das Applizieren oben erwähnter Kurzschlüsse physikalisch defekt ist.

Prinzipiell gibt es für Eingriffe in die inneren Prozesse des Routers im Ernstfall zwei Möglichkeiten: Der Zugriff über die serielle Schnittstelle (beim WRT ist eine vorhanden) muss erfolgen, solange wenigstens der Bootloader noch funktioniert. Wenn auch dieser nicht mehr ansprechbar ist, hilft nur noch ein Programmierkabel für den direkten Hardwarezugriff per JTAG-Schnittstelle, sofern das Gerät darüber verfügt.

JTAG verursacht die geringsten Kosten und ermöglicht alle erforderlichen Aktionen zur Systemherstellung. Alle hier beschriebenen Geräte besitzen diesen Anschluss. Ein dazu passendes JTAG-Kabel selbst zu löten ist für Elektronikbastler keine große Hürde und kostet auch nicht viel. Hat man ein solches Kabel in der Schublade, gestaltet sich das Flashen sehr viel entspannter.

JTAG-Kabel anfertigen

Ein einfaches, passives JTAG-Kabel lässt sich aus einem Stecker für den Druckerport eines PC (25poliger Stecker, Sub-D), vier Widerständen zu 100 Ohm und einem zweireihigen weiblichen Pfostenstecker im 2,54-mm-Raster mit 12 Pins und passendem Flachbandkabel herstellen. Die Bauteile findet man im gut sortierten Elektronikgeschäft oder in der Bastelkiste. Den Stecker für den Druckerport nebst Anschlusskabel kann man aus einem alten Druckerkabel gewinnen.

Einen zweireihigen weiblichen Pfostenstecker mit 10 Pins findet man an einem Adapterkabel aus einem alten PC, mit dem beispielsweise die serielle Schnittstelle vom Mainboard nach außen zum Slotblech geführt wird. Bei

WRT-Geräten hat der JTAG-Anschluss 12 Pins, die Pins 1 und 2 sind jedoch für den Anschluss von JTAG nicht erforderlich. Also kommt man mit einem 10-poligen weiblichen Pfostenstecker aus, wenn auf dem Routerboard die Pins 1 und 2 nicht bestückt sind.

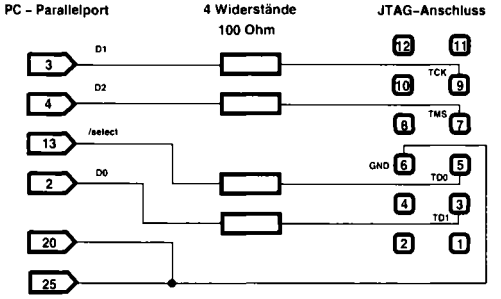


Abbildung 6.8: Schaltplan JTAG-Kabel

Das JTAG-Kabel darf nicht zu lang sein, da es sonst zu Übertragungsfehlern kommen kann. Mit einer Länge von maximal 70 Zentimetern ist man auf der sicheren Seite. Da sich beim Zusammenbau des Kabels leicht Fehler einschleichen, sollte man den Adapter vor dem ersten Einstecken gründlich mit einem Multimeter prüfen, um den Druckerport nicht zu beschädigen. Idealerweise testet man den Adapter zuerst an einem alten PC.

Ist das JTAG-Kabel fertig, muss noch der dazugehörige JTAG-Anschluss auf dem Mainboard des Routers mit einer zweireihigen Pfostenleiste im Raster 2,54 Millimeter bestückt werden. Die Pfostenleiste bekommt man stangenweise als einreihige oder zweireihige Ausführung in Elektronikgeschäften. Sie wird auf die gewünschte Länge gekürzt und kann doppelt ausgeführt werden, falls nur einreihige Pfostenleisten verfügbar sind. Zum Löten wird ein Lötkolben mit feiner Spitze und etwa 20 Watt Leistung und etwas Elektroniklot (kein Fittingslot, Lötfett oder ähnliches verwenden) benötigt. Nach dem Programmieren verbleibt die Pfostenleiste auf dem Router, sie kostet nur einige Cent.

Abbildung 6.9 zeigt das Innenleben eines WRT54GL Revision 1.1. Die Spitze des Schraubendrehers zeigt auf den unbestückten JTAG-Anschluss, darüber der ebenfalls unbestückte Anschluss für die serielle Schnittstelle, der ebenfalls mit einer Pfostenleiste bestückt werden kann. Dazu wird jedoch ein RS232-Pegelwandler benötigt[12]. Der Router sollte nur im ausgeschalteten Zustand an das JTAG-Kabel angeschlossen werden. Wie immer gilt bei derartigen Basteleien: Auf eigene Gefahr und eigenes Risiko!

[12] Nähere Auskünfte zur seriellen Schnittstelle sind im Wiki von `http://openwrt.org` zu finden.

Abbildung 6.9: WRT54-Router mit JTAG-Anschluss

Das JTAG-Programm `wrt54g`

Für Windows, FreeBSD und Linux gibt es das JTAG-Programm `wrt54g`[13] unter GNU/GPL-Lizenz. Es ist für die Anwendung am WRT54G/GL mit dem beschriebenen passiven JTAG-Kabel gedacht und funktioniert zuverlässig. Allerdings ist das Schreiben per JTAG mit dem einfachen Kabel sehr langsam. Das Programmieren eines neuen Bootloaders mit 256 kByte Größe dauert annähernd 26 Minuten. Voraussetzung für die Arbeit mit dem JTAG-Programm ist der Zugriff auf den Parallelport des PC. Leider ist es der Autorin nicht gelungen, den Parallelport unter Windows 2000 mit dem Programm anzusprechen. Unter Linux funktionierte es jedoch auf Anhieb. Sollten unter Linux dennoch Probleme mit dem Parallelport auftauchen, kann man diese Checkliste durchgehen:

- Sind die Kernelmodule `parport` und `parport_pc` für den Parallelport des PC geladen? Die Antwort gibt `lsmod | grep parport`. Im Bedarfsfall muss man sie mit `modprobe parport_pc` laden.
- Ist der Parallelport bereits durch das Kernelmodul `lp` belegt, wird es mit `rmmod lp` entladen.
- Zeigt `ls /dev/parport0`, dass die Gerätedatei `/dev/parport0` existiert? Fehlt sie, kann sie mit

  ```
  mknod /dev/parport0 c 99 0 -m 666
  ```

  angelegt werden. Dann sollte man die Kernelmodule mit `rmmod parport parport_pc` noch einmal entladen und mit `modprobe parport_pc` anschließend neu laden.

[13] `http://downloads.openwrt.org/utils/HairyDairyMaid_WRT54G_Debrick_Utility_v48.zip`

Zur Arbeit mit `wrt54g` wechselt man auf der Kommandozeile in das Verzeichnis, in dem sich das Programm befindet, und startet es mit vorgestelltem Punkt und Schrägstrich (Slash) als Pfadangabe. Der Befehl `./wrt54g` ohne weitere Startparameter gibt eine Hilfedatei aus.

NVRAM mit JTAG löschen

Verhindern falsche Einstellungen im NVRAM den erfolgreichen Start des Systems, kann das NVRAM einfach per JTAG gelöscht werden. Das genügt häufig schon, um den Router wieder ansprechbar zu machen. Um die anschließende Wiederherstellung des NVRAM-Inhalts braucht man sich nicht zu kümmern. Findet der CFE ein leeres oder beschädigtes NVRAM vor, schreibt er automatisch das NVRAM neu und trägt die Fabrikeinstellungen ein. Diese sind im CFE gespeichert. Sicherheitshalber sollte vorher ein Backup angelegt werden:

```
./wrt54g -backup:nvram
./wrt54g -erase:nvram
```

Falls das NVRAM auf zuvor gesicherte – funktionierende – Einstellungen zurückgesetzt werden soll kann, per JTAG ein Backup eingespielt werden. Das Flashen des NVRAM dauert etwa sechs Minuten:

```
./wrt54g -flash:nvram
```

Kernel mit JTAG löschen

Für eine Havarie der Firmware hat Linksys eine Hintertür vorgesehen: Ist der Kernel beschädigt, bleibt der Bootprozess stehen und der CFE aktiviert den TFTP-Server. Der erwähnte grobe Trick mit dem Kurzschluss am Flashchip nutzt dies aus. Er bewirkt, dass der Bootloader den Inhalt des Flashspeichers als beschädigt erkennt und den TFTP-Server aktiviert. Hat der Router diesen Hack heil überstanden und wurde beim Kurzschließen zufällig der richtige Moment erwischt, kann per TFTP-Upload ein lauffähiges System installiert werden. Risikoärmer und erfolgversprechender ist es, den Kernel mit dem JTAG-Kabel einfach zu löschen:

```
./wrt54g -erase:kernel
```

Beim nächsten Neustart bleibt der Bootloader nun stehen und wartet geduldig auf ein TFTP-Upload. Für die initiale Firmwareinstallation nach dem Einsatz der JTAG-Kabels sollte zuerst die Originalfirmware von Linksys verwendet werden. Die Firmware-Version muss genau zu der vorhandenen Hardwarerevision passen. Idealerweise nimmt man exakt die Version, mit

der der Router ausgeliefert wurde. Bei einem praktischen Test gelang das Wiederherstellen des Systems nur mit der deutschen Originalfirmware auf einem in Deutschland gekauften Gerät. Eine amerikanische Version der Linksys-Firmware mit einer höheren Versionsnummer funktionierte nicht.

Nachdem die Linksys-Firmware erfolgreich installiert ist, kann problemlos wieder eine alternative Firmware geflasht werden.

Bootloader defekt

Wenn alle Ethernet-LEDs dauerhaft leuchten und dabei die Power-LED permanent blinkt, ist der Bootloader CFE defekt oder gelöscht. Per JTAG kann ein neuer Bootloader installiert werden. Der CFE ist bei jedem einzelnen WRT individuell unterschiedlich, da er die MAC-Adresse des Ethernetinterfaces enthält. Außerdem unterscheiden sich die Bootloader bei verschiedenen Modellreihen und Hardwarerevisionen. Im Internet findet man Webseiten, die online eine individuell auf das Gerät zugeschnittene Bootloaderdatei erzeugen oder geeigldenete Hilfsprogramme zum kostenlosen Download bieten[14].

Mehrere Geräte mit exakt der gleichen Bootloaderdatei mit gleicher MAC-Adresse zu beschrieben ist keine gute Idee, obwohl das möglich ist. Eine MAC-Adresse sollte immer ein Unikat sein. Wenn doppelte MAC-Adressen in einem Netzwerk vorkommen, wird das sehr wahrscheinlich zu Netzwerkproblemen mit dem ARP-Protokoll führen. Folgender Befehl flasht den Bootloader:

```
./wrt54g -flash:cfe
```

Anschließend kann noch der Kernel gelöscht werden, um an den TFTP-Server heranzukommen:

```
./wrt54g -erase:kernel
```

Nach dem Neustart bootet der Router in den TFTP-Server. Nun sollte zuerst die Originalfirmware von Linksys installiert werden (siehe Abschnitt 6.2.1).

[14] Einen CFE-Onlinegenerator für den WRT gibt es unter der Adresse `http://lonewolf.hacker-nin.com/wrt/cfe/cfe.php`. Die deutschsprachige Website `http://www.wlan-skynet.de` bietet mit dem „Skynet-Repair-Kit" für den WRT eine Sammlung von Softwaretools für Windows-User. Mit dabei ist ein TFTP-Programm, ein Bootloader-Generator und das JTAG-Programm `wrt54g`.

Kapitel 7

Elektromagnetische Wellen und Antennen

Die meisten Hersteller geben für ihre SoHO-Geräte eine Reichweite von wenigen hundert Metern im Freien oder weniger als 100 Meter innerhalb von Gebäuden an. Mit etwas Basiswissen über die Physik der elektromagnetischen Wellen und zusätzlichen Antennen lässt sich die Reichweite jedoch auf mehrere Kilometer steigern, ohne dass die Übertragungsgeschwindigkeit dabei drastisch leidet.

Der derzeitige Weltrekord für die längste funktionierende WiFi-Funkstrecke mit einem herkömmlichen SoHO-Accesspoint (Linksys WRT54GL) beträgt 382 Kilometer bei einer Sendeausgangsleistung von 0,1 Watt. Über die Funkstrecke konnte das Team von Ermanno Pietrosemoli aus Venzuela Videotelefonie betreiben, die Bandbreite betrug immerhin noch 65 kBit/Sekunde. Damit hat das Team seinen alten Rekord von 279 Kilometer eingestellt. Durch Veränderungen am MAC-Protokoll von 802.11 war es ihnen möglich, rund 3 MBit/Sekunde über ihre alte Rekorddistanz zu übertragen. Doch bevor wir dazu kommen, wie das möglich ist, ist etwas Theorie unumgänglich.

7.1 Bel und Dezibel

In der Hochfrequenztechnik treten oft gigantische Größenverhältnisse auf. Der Umgang damit wird durch Vergleichsgrößen erleichtert. Das Bel ist eine logarithmische Hilfsgröße zum Vergleich von Messgrößen. Es ist definiert als der dekadische Logarithmus des Quotienten zweier Leistungen. Ein *Dezibel*, also ein Zehntel Bel, ist die gebräuchlichste Vergleichsgröße im Bereich der Hochfrequenztechnik und der Akustik. Die gebräuchliche Schreibweise für Dezibel ist *dB*. Das Verhältnis v zweier Leistungen P_2 und P_1 in dB ergibt sich also aus:

$$v(dB) = 10 * log_{10} \frac{P_2}{P_1}$$

Größenvergleiche und -berechnungen können mit dem Dezibel durch einfaches Addieren und Subtrahieren der Dezibelwerte durchgeführt werden, anstatt die tatsächlichen physikalischen Werte zueinander ins Verhältnis zu setzen.

Positive Größenverhältnisse (Verstärkung) haben ein positives Vorzeichen, negative Größenverhältnisse (Reduzierung, Dämpfung) entsprechend ein negatives Vorzeichen. Damit lassen sich komplizierte Berechnungen wunderbar ausführen, wie gleich zu sehen sein wird.

Die Hersteller geben die Sendeleistung von WiFi-Karten meist nicht in Watt oder Milliwatt an, sondern in dBm. Man beachte das kleine m nach dem dB. Die Leistungsgröße dBm bedeutet: im Vergleich zu 1 mW, P_1 beträgt also immer ein Milliwatt.

Bei einem Verhältnis von 1, also einer Leistung von 1 mW, ist der Logarithmus 0, wir haben also eine Verstärkung (oder auch einen *Pegel*) von 0 dB. Ist die Leistung doppelt so groß wie 1 mW, ergibt sich

$$3{,}0103 = 10 * log_{10}(2/1)$$

Man kann sich also leicht sich merken, dass 3 dB immer recht genau einer Verdopplung, also dem Faktor 2 entsprechen. Bei einer Einheit von dBm entsprechen also 3 dB einer Leistung von 2 mW, 6 dB wären 4 mW und so weiter.

Jeder Schritt von 10 dB entspricht genau einer Verzehnfachung des zugrunde gelegten Milliwatts: 10 dBm sind also 10 mW, 20 dBm sind 100 mW, 30 dBm sind 1 Watt.

Am einfachsten rechnet es sich natürlich mit der Logarithmus-Taste auf dem Taschenrechner. Allerdings rechnet der Taschenrechner in Bel und nicht in Dezibel. Deshalb werden die Werte nach dem Logarithmieren mit 10 multipliziert bzw. vor dem Delogarithmieren (Invers- und Log-Taste) durch 10 dividiert.

Um die Sendeleistung einer WiFi-Karte mit 330 mW in dBm umzurechnen, genügt es, auf dem Taschenrechner 330 einzutippen und die Log-Taste zu drücken. Als Resultat erhält man 2,518 Bel, die 330 mW entsprechen also 25,18 dBm.

Ein weiteres Beispiel: Angenommen, der Gewinn einer Antennenschüssel von 2 Metern Durchmesser soll 32 dB sein. Um welches Verhältnis verstärkt sich das Signal? Die Lösung: 3,2 eintippen, Invers-Taste drücken, Log-Taste drücken. Das Resultat ist 1584,9.

7.2 Das Phänomen der elektromagnetischen Wellen

Ein elektrisches Kabel, das von Gleichstrom durchflossen wird, erzeugt ein magnetisches Kraftfeld quer zum Leiter. Gleichzeitig befindet sich ein elektrisches Spannungsfeld längs zum Kabel, das den Strom durch den Leiter treibt. Das magnetische Feld kann leicht nachgewiesen werden, wenn man eine Kompassnadel in die Nähe eines stromdurchflossenen Leiters bringt. Die Kompassnadel wird von dem magnetischen Kraftfeld des elektrischen Stroms abgelenkt.

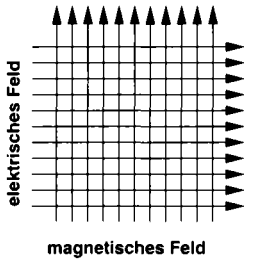
Fließt der Strom gleichmäßig, bildet sich um das stromdurchflossene Anschlusskabel ein ruhendes magnetisches Feld wie bei einem Dauermagneten. Dreht man die Polung der Spannungsquelle um, ändert der Strom die Flussrichtung. Dadurch ändert auch das magnetische Feld seine Polarität.

Jedes Mal, wenn eine Veränderung im Stromfluss des Stromkreises eintritt (der Strom wird ein- oder ausgeschaltet, die Polarität wird umgedreht), lösen sich das elektrische und magnetische Feld zusammen von dem Leiter ab und breiten sich mit Lichtgeschwindigkeit im Raum aus. Die Analogie zu Wasserwellen drängt sich an diesem Punkt auf – wirft man einen Stein ins Wasser, erzeugt dieser auf der Wasseroberfläche eine Wellenbewegung, die sich ausbreitet. Erzeugt man auf der Wasseroberfläche kontinuierlich Bewegung, breiten sich auch kontinuierlich Wellen aus. Schaltet man einen Stromkreis ein, erzeugt man ebenfalls eine Wellenbewegung unsichtbarer elektromagnetischer Felder. Findet in einem Stromkreis eine kontinuierliche Veränderung statt, entsteht eine kontinuierliche Wellenbewegung, die sich vom Ort ihrer Entstehung ausbreitet.

Wechselstrom wechselt im Gegensatz zu Gleichstrom beständig die Polarität. Schließt man an den Stromkreis Wechselstrom an, entstehen kontinuierlich elektromagnetische Wellen, die sich im Raum mit Lichtgeschwindigkeit ausbreiten. Nach diesem grundlegenden Prinzip – dem Erzeugen eines Wechselstroms – arbeiten alle Sender, die elektromagnetische Wellen produzieren.

Elektromagnetische Wellen haben je ein elektrisches und ein magnetisches Kraftfeld, die beide im Winkel von 90 Grad zueinander stehen und die man durch Feldlinien darstellen kann (Abbildung 7.1). Beide Felder sind bipolar. Das elektrische Feld ist positiv und negativ polarisiert, beim magnetischen Feld spricht man hingegen vom Nord- und Südpol.

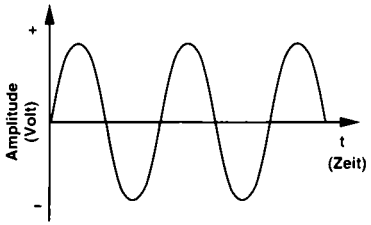
Abbildung 7.1: Feldlinien



Elektromagnetische Wellen sind eine Form von Energie, die sich von ihrem Entstehungsort aus im Raum mit Lichtgeschwindigkeit ausbreitet. Elektromagnetische Energie benötigt lediglich Raum, um sich auszubreiten – ein Übertragungsmedium ist nicht erforderlich. Als die elektromagnetischen Wellen am Ende des 19. Jahrhunderts entdeckt wurden, glaubte man anfangs noch, dass ein bisher unentdecktes Übertragungsmedium dafür erforderlich sei, das man „Äther" taufte. Dies stellte sich als Irrtum heraus, trotzdem hat sich der Begriff Äther als nicht ganz ernst gemeinte Bezeichnung für das Medium Funk erhalten. Im All, das überwiegend nur leerer Raum ist, können sich elektromagnetische Wellen über enorme Entfernungen ausbreiten. Die Radioastronomie nutzt diese Eigenschaft aus.

Charakteristisch für elektromagnetische Wellen sind ihre Amplitudenhöhe, Wellenlänge und ihre Frequenz. Die Skizze in Abbildung 7.2 betrachtet nur eine Komponente der elektromagnetischen Wellen – das elektrische Feld.

Abbildung 7.2: Sinuskurve mit Amplitude und Zeit



Die Grafik zeigt eine Welle, die zweieinhalb Zyklen innerhalb des dargestellten Zeitabschnitts absolviert. Wellen unterscheiden sich in ihrer Schwingfrequenz, also der Anzahl von Zyklen, die eine Wellenfront innerhalb einer definierten Zeit durchläuft. Die Einheit für die Frequenz trägt den Namen Hertz und gibt an, wie viele Schwingungen (Zyklen) eine elektromagnetische Welle in einer Sekunde durchläuft. Durchläuft eine Schwingung einen Zyklus pro Sekunde, ist die Frequenz ein Hertz (abgekürzt Hz).

Die Wellenlänge gibt an, wie weit sich eine elektromagnetische Wellenfront im Raum ausbreitet, bis sie eine einzelne Schwingung abgeschlossen hat. Da sich elektromagnetische Wellen mit Lichtgeschwindigkeit im Raum ausbreiten, legen sie in einer Sekunde 300 000 Kilometer zurück (um ganz genau zu sein: 299 792 458 Kilometer). Also muss die Wellenlänge bei einer Schwingung pro Sekunde (1 Hz) 300 000 Kilometer betragen.

Als Symbol für die Wellenlänge wird das Zeichen Lambda λ verwendet. $\lambda/2$, das für die Gestaltung von Antennen und Verstärkern eine wichtige Rolle spielt, bedeutet also „halbe Wellenlänge".

WiFi nach IEEE 802.11 b und g verwendet einen wesentlich höheren Frequenzbereich, nämlich zwischen 2,405 und 2,480 GHz, also Frequenzen zwischen 2,405 und 2,480 Milliarden Schwingungen pro Sekunde.

Entsprechend beträgt die mittlere Wellenlänge auch nicht mehr 300 000 Kilometer – wie bei der einleitenden 1-Hertz-Schwingung – sondern lediglich noch:

$$\lambda = 0,125\,m = \frac{300000000\,m/s}{2440000000}$$

also gerade mal 12,5 Zentimeter. Der IEEE-Standard 802.11a verwendet einen noch höheren Frequenzbereich im elektromagnetischen Spektrum zwischen 4,9 und 5,9 GHz. Bei diesen Frequenzen beträgt die mittlere Wellenlänge λ nur noch 5,5 cm.

Zum Vergleich: Die Wellenlänge eines UKW-Radiosenders bei 100 MHz beträgt 3 Meter, ein Mobiltelefonnetz nach dem Standard GSM1800 (wie es beispielsweise von E-Plus oder O2 verwendet wird) hat eine Wellenlänge von 17,5 bis 16,2 Zentimetern. Das menschliche Auge kann Wellenlängen von 711 nm (rot) bis 389 nm (violett) wahrnehmen, denn auch das Licht gehört zu den elektromagnetischen Wellen.

7.3 Das Prinzip von Antennen

Wechselstrom erzeugt also in einem Leiter elektromagnetische Wellen, die an den Raum abgegeben werden; aber dieses Prinzip gilt auch umgekehrt: Treffen elektromagnetische Wellen auf einen elektrischen Leiter, erzeugen

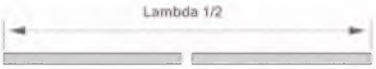
sie in diesem Wechselstrom. Für jedes Verhältnis von Länge und Oberfläche eines elektrischen Leiters tritt bei einer bestimmten Frequenz ein Resonanzeffekt auf – der Leiter wird bei seiner Resonanzfrequenz wie eine gespannte Gitarrensaite zu starken Schwingungen angeregt. Der Leiter arbeitet dann als Antenne für seine Resonanzfrequenz. Am besten funktioniert die Übertragung von elektromagnetischen Wellen über Antennen, wenn Sende- und Empfangsantenne auf die richtige Resonanz abgestimmt sind.

Antennen sind stets reziprok, das heißt sie können sowohl senden als auch empfangen. Antennen, die nur senden oder empfangen können, gibt es prinzipiell nicht, ausgenommen Antennen mit eingebauten elektronischen Verstärkerschaltungen. Dann liegt es aber an der Elektronik und nicht an der Antenne.

Die elektrische Energie, die durch das „Einfangen" von elektromagnetischen Wellen in einer Antenne ankommt, ist beim Empfangen natürlich viel geringer als beim Senden[1]. Mit elektronischen Verstärkerschaltungen lassen sich jedoch auch schwache elektromagnetische Felder über große Entfernungen nachweisen und auswerten, solange das Signal eine bestimmte Mindeststärke hat und nicht von Stör- oder Rauschsignalen zugedeckt wird.

Einer der bedeutendsten Pioniere der Hochfrequenztechnik war der Physiker Heinrich Rudolf Hertz, der die Grundform der meisten Antennen entdeckt hat, den Hertzschen Halbwellendipol, oder kurz Dipol. Hertz entdeckte, dass ein Leiter bei halber Wellenlänge in Resonanz gerät. Der Dipol ist das einfachste Resonanzgebilde, er bildet das Grundelement der meisten Antennen und wird bei Messungen als Referenzantenne zum Vergleich des Antennengewinns herangezogen. Üblicherweise ist der Halbwellenleiter in seiner geometrischen Mitte aufgetrennt. An den so entstehenden beiden Anschlusspolen (daher stammt die Bezeichnung Dipol) kann die Speiseleitung von Sender/Empfänger angeschlossen werden.

Abbildung 7.3: Dipol



In den meisten WLAN-Karten oder Stummelantennen von Accesspoints befinden sich Dipole. Die Dipole in den Stummelantennen der Accesspoints sind meistens sogenannte Sleeve-Dipole (von engl. Ärmel), bei denen die Speiseleitung durch einen Schenkel der Antenne (ein Röhrchen) wie durch einen Ärmel zum Speisepunkt in der Antennenmitte geführt wird.

[1] Trotzdem hatte Nikola Tesla, der Pionier der Wechselstromtechnik, die Idee, man könne die Welt drahtlos mit Energie versorgen. Elektromagnetische Wellen können aber nur geringe Energiemengen auf vergleichsweise kurze Entfernungen drahtlos übertragen. Bei RFID-Etiketten und Transpondern macht man sich diesen Effekt zunutze.

Dipolantennen sind Rundstrahler, die bevorzugt kreisrund zur Seite abstrahlen und weniger stark zu den Enden hin. Man kann sich das Strahlungsdiagramm in der Form eines amerikanischen Donut vorstellen.

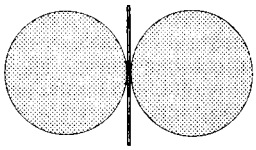


Abbildung 7.4: Strahlungsdiagramm eines Dipols

Weil Dipole die Energie eher zur Seite abstrahlen beziehungsweise von der Seite besser aufnehmen, resultiert daraus ein kleiner Antennengewinn gegenüber einer (nur theoretisch existierenden) Antenne, die kreisrund in alle Richtungen abstrahlt.

7.3.1 Polarisation von Antennen

Steht die Stummelantenne eines Accesspoints senkrecht zur Erdoberfläche, steht auch das von der Antenne beim Senden abgestrahlte elektrische Feld senkrecht oder vertikal. Man sagt daher, dass die Antenne vertikal polarisiert ist, und bezieht sich dabei auf die elektrische Komponente der elektromagnetischen Wellen. Stellt man die Stummelantenne horizontal, ist die Antenne entsprechend horizontal polarisiert. Beim Aufstellen von Antennen muss diesem Punkt Beachtung geschenkt werden, da sich zwei Antennen kaum gegenseitig empfangen können, wenn sie eine um 90 Grad gegeneinander gedrehte Polarisation haben.

Die Stärke des Empfangssignals einer horizontal polarisierten Antenne gegenüber einer vertikal polarisierten wird um den Faktor 100 abgeschwächt. Das Drehen der Polarisationsebene bewirkt also einen enormen Signalverlust von 20 Dezibel. In der Praxis wirken sich falsche Polarisationsebenen nicht immer so drastisch aus, da sich die Polarisationsebenen der Wellenfronten durch Reflektionen teilweise drehen können.

Horizontale und vertikale Polarisation werden als lineare Polarisation bezeichnet. Es gibt aber auch Antennen – wie beispielsweise Helical-Antennen oder Patch-Antennen –, die ein rotierendes elektromagnetisches Feld erzeugen. Das wird als zirkulare Polarisation bezeichnet. Zirkulare Polarisation kann sowohl rechtsdrehend als auch linksdrehend sein. Werden linear polarisierte Signale mit einer zirkular polarisierten Antenne empfangen (und umgekehrt), ergibt sich immer ein Signalverlust von 3 Dezibel. Dabei spielt es dann aber keine Rolle mehr, in welchem Winkel ein eintreffendes linear polarisiertes Signal bei einer zirkularen Antenne gerade eintrifft. Drastischer sind die Verluste, wenn eine linksdrehende zirkular polarisier-

te Antenne mit einer rechtsdrehenden Antenne kommunizieren soll: Diese sind für einander praktisch blind. Daher verwendet man aus Konvention überwiegend rechtsdrehende zirkulare Polarisation bei Antennen.

Bei der Aufstellung von Antennen an exponierten Standorten kann die bewusste Auswahl der Polarisationsebene deutliche Vorteile bringen – wenn man nämlich eine Polarisationsebene wählt, die eben nicht von den meisten konkurrierenden Anwendungen benutzt wird. Die meisten Anwender richten die Stummelantennen ihrer Accesspoints wahrscheinlich senkrecht aus. Es kann daher vorteilhaft sein, Antennen horizontal polarisiert aufzustellen, um Störungen von Antennen mit vertikaler Polarisation zu unterdrücken.

7.3.2 Freifelddämpfung und Absorptionsverluste

Wenn sich elektromagnetische Wellen von einem Ausgangspunkt im Raum ausbreiten, nimmt mit größer werdendem Abstand von der Quelle die Energie ab. Der Grund dafür: Die Fläche, über die sich die elektromagnetische Energie verteilt, wird größer.

Angenommen der Ausgangspunkt einer elektromagnetischen Wellenfront ist ein sehr kleines, kugelförmiges Objekt, welches in alle Richtungen gleichmäßig strahlt. Bei einem Meter Abstand hat sich die Energie der Wellenfront bereits über eine kugelförmige Fläche mit einem Durchmesser von zwei Metern verteilt.

Nach der Formel für die Kugeloberfläche:

$$A_O = 4 * \pi * r^2$$

verteilt sich die elektromagnetische Energie bei einem Abstand von einem Meter auf eine Fläche von 12,56 m^2. Bei zwei Metern Abstand verteilt sich die elektromagnetische Energie auf eine Fläche von 50,27 m^2.

Die elektromagnetische Strahlungsdichte nimmt also mit dem Quadrat zur Entfernung ab. Die Signaldämpfung (Freifelddämpfung) zwischen einer sendenden und einer empfangenden Antenne wird in dB angegeben und ist frequenzabhängig. Das liegt daran, dass die empfangende Antenne eine Antennenwirkfläche hat, die von der Wellenlänge abhängig ist. Je größer die Antennenwirkfläche, desto mehr „fischt" die Antenne von dem im Raum verteilten Signal heraus. Die Formel zur Berechnung der Freifelddämpfung D in dB für die Entfernung d lautet:

$$D(dB) = 20 * log_{10}(4 * \pi * d/\lambda)$$

Bei Regen, Schnee und Nebel steigen auf einer Funkstrecke die Verluste durch Absorption. Bei 5,8 GHz muss man mit zusätzlichen Verlusten von

0.5 dB pro Kilometer Funkstrecke rechnen, bei 2.4 GHz kann man von nur 0.1 dB/km ausgehen.

Objekte, die sich als Hindernisse im Ausbreitungsgebiet von elektromagnetischen Wellen befinden, sind umso störender, je größer sie im Verhältnis zur Wellenlänge der elektromagnetischen Energie sind. Eine Welle von der Länge eines Wolkenkratzers lässt sich kaum von einem Objekt von der Größe einer Streichholzschachtel in ihrer Ausbreitung stören – umgekehrt dagegen schon.

Da die Wellenlängen von WiFi nur einige Zentimeter betragen, gibt es viele Objekte, an denen einer oder mehrere von folgenden Effekten auftreten:

Abschattungen
: Ein für die Wellenfront undurchdringliches oder stark dämpfendes Objekt blockiert die Ausbreitung, so dass dahinter kein Empfang möglich ist.

Reflektionen
: Die Wellenfront wird gespiegelt.

Streuung
: Eine Wellenfront wird beim Auftreffen auf ein Objekt in mehrere schwächere Wellenfronten aufgeteilt.

Beugung
: Beim Auftreffen auf die Kante eines Objekts ändert sich die Ausbreitungsrichtung der Wellenfront.

Trockene, nicht leitende Materialien stören die Ausbreitung von elektromagnetischen Wellen kaum (z. B. eine Rigipswand). Dagegen sind gut leitende Materialien praktisch undurchdringlich und reflektieren die Wellenfront (etwa eine dicke feuchte Mauer). Bäume sind für WLAN-Wellen nahezu undurchdringlich, vor allem wenn sie im vollen Saft sind. Das soll nicht heißen, dass man nicht auf kurze Entfernungen durch einen einzelnen Baum oder eine Reihe Alleebäume funken kann – die Signale werden allerdings stark gedämpft.

7.4 Antennengewinn

Viele Anwender haben den unbedarften Glauben, dass eine gewinnbringende Antenne elektromagnetische Energie auf magische Weise vermehren kann – was natürlich nicht der Fall ist. Antennengewinn kommt nicht pauschal durch eine Verstärkung von Signalen zustande, sondern lediglich durch Bündelung der vorhandenen Energie beim Senden und Empfang in einer bestimmten Ebene oder Hauptstrahlrichtung.

Wenn eine Antenne zu anderen Teilnehmern funken soll, die in allen Richtungen im Raum verteilt sind, ist die Installation einer gewinnbringenden Antenne kontraproduktiv. Wenn also ein Accesspoint in der Mitte eines Hauses steht und das ganze Haus versorgen soll, ist die mit dem Accesspoint mitgelieferte Stummelantenne die beste Antenne für den Zweck.

Eine Antenne, die kugelförmig in alle Richtungen gleichmäßig strahlt, gibt es in der Praxis nicht. Trotzdem wird diese rein theoretische Antenne, die als Isotrop-Strahler bezeichnet wird, für Vergleichswerte als Referenz verwendet, da sie keinerlei Antennengewinn aufweist. Antennengewinne werden oft als dBi-Werte angegeben. Der Antennengewinn in dBi gibt den Grad der Verstärkung in Relation zum theoretischen Kugelstrahler an.

Eine Antenne mit Gewinn gegenüber der isotropen Antenne konzentriert die elektromagnetische Energie in einer bestimmte Richtung. Je stärker die elektromagnetische Energie gebündelt wird, desto höher ist bei einer gut gebauten Antenne der Gewinn. Antennengewinne werden auch – weitaus realistischer – in Relation zu einem Dipol angegeben (dBd). Da eine Dipolantenne jedoch einen Gewinn von 2,2 dB gegenüber einer isotropen Antenne aufweist, sind Gewinnangaben in dBd um 2,2 dB kleiner als in dBi und damit weniger werbewirksam.

Ein seriöser Hersteller gibt dB-Werte in dBi oder dBd an. Eine verbreitete Unsitte ist eine Angabe in dB. In diesem Fall kann man davon ausgehen, dass der Hersteller dBi meint, falls seine Angaben überhaupt fundiert sind. Unseriöse Hersteller überteiben gerne bei den Gewinnangaben, da sich höhere Werte besser verkaufen. Bei allzu tollen Gewinnangaben sollte man skeptisch sein. Verdächtig ist auch, wenn in Datenblättern Strahlungsdiagramme auftauchen, die exakt symmetrisch aussehen und keine unregelmäßigen Nebenzipfel aufweisen – diese sind phantasiert und nicht gemessen.

7.5 Antennentypen

Prinzipiell gibt es drei Antennentypen: Rundstrahlende Antennen, Sektorantennen und Richtantennen.

7.5.1 Omni-Antennen

Rundstrahlende Antennen (man nennt sie auch Omnis, wegen der englischen Bezeichnung Omnidirectional Antennas) haben in vertikaler Ebene ein kreisrundes Strahlungsdiagramm. Ein vertikal montierter Dipol ist eine einfache rundstrahlende Antenne. Der vertikale Dipol ist keine Antenne mit ausgeprägtem Gewinn, da das Strahlungsdiagramm in der horizontalen Ebene sehr breit ist: Es geht relativ viel Energie nach unten und nach oben.

In der Regel ist das unerwünscht. Höhere Reichweiten lassen sich erzielen, wenn die Strahlungsenergie stärker zur Seite wirkt anstatt nach oben oder unten.

Diesen Effekt, das Strahlungsmaximum weiter zur Seite zu konzentrieren, erreicht man in der Regel durch die Übereinanderstockung von mehreren Dipolen. Allerdings schätzen viele Anwender den Nutzen von stark gewinnbringenden Rundstrahlantennen falsch ein: Allzu stark in horizontaler Ebene bündelnde Omnis funken von hohen Gebäuden in die weite Ferne und empfangen gleichzeitig auch viele weit entfernte Störungen, aber die WLAN-Knoten am Boden unterhalb der Strahlungskeule bekommen davon nur wenig mit.

Gerade bei Omnis sollte man gegenüber utopischen Gewinnangaben der Hersteller und Verkäufer eine gesunde Skepsis walten lassen. Da Omnis vertikal 360 Grad abdecken müssen, können große Gewinne nur durch extrem scharfe Bündelung in horizontaler Ebene erzielt werden. Gewinnangaben jenseits von 10 dBi sind wahrscheinlich Unsinn. Die wenigsten scharf bündelnden Omnis strahlen wirklich waagrecht zur Seite – nicht wenige strahlen stattdessen in einem Winkel von einigen Grad nach oben, so dass sie eigentlich bei der Montage auf den Kopf gestellt werden müssten. Von einer derartigen Antenne hat man nur wenig.

Eine wirklich gute Omni strahlt in einem Winkel von wenigen Grad nach unten. Gut funktionierende Omnis lassen sich nur mit einigem Aufwand selbst bauen. Auf dem Markt gibt es oft billige Angebote mit fantastischen Gewinnangaben, die ihr Geld nicht wert sind. Im direkten Vergleich mit der mitgelieferten Stummelantenne des Accesspoints stellt sich schnell Ernüchterung ein, zumal Verluste in Antennenkabel und Steckverbindern vom Gewinn der Antenne abgezogen werden müssen. Bei der Montage einer stark bündelnden Omni sollte der Strahler genau lotrecht montiert werden, da sonst in einer Richtung zum Himmel gefunkt wird.

7.5.2 Sektorantennen

Sektorantennen sind häufig auf den Sendemasten der Mobiltelefonanbieter zu sehen. Diese Antennen sind überwiegend dazu gedacht, um von einem exponierten Standort aus einen Bereich in mehrere Sektoren zu unterteilen, der dann von mehreren Funksystemen versorgt wird. Man kann also beispielsweise ein Gebiet in vier Sektoren mit 90 Grad aufteilen und jeden Sektor mit einem SoHO-Router abdecken.

Sektorantennen bündeln horizontal sehr stark und haben eine sehr breite vertikale Strahlungskeule. Da sie horizontal stark bündeln, werden sie an hohen Standorten um ein paar Grad nach unten geneigt montiert. Die meisten Sektorantennen sind von ihrem Aufbau her omnidirektionale Antennen vor einer Reflektorwand.

7.5.3 Richtantennen

Richtantennen verengen das Strahlungsdiagramm in horizontaler und vertikaler Ebene. Ihr Sinn ist es, stabile und leistungsfähige Punkt-zu-Punkt-Verbindungen über große Entfernungen zu erzielen. Am zuverlässigsten und leistungsstärksten sind Richtantennen mit parabolischen Reflektoren, die gemeinhin als Schüsseln (englisch: Dish) bezeichnet werden. Professionelle Richtantennen mit großem Gewinn verwenden als Reflektoren Gitter aus Aluminiumdruckguss oder Stahlrohren, um die Windlast zu senken. Einfache Richtantennen (Dosenantenne und Biquad-Antenne) können mit einfachen Mitteln selbst hergestellt werden[2].

7.6 Wireless Long Shots - Funkstrecken über große Entfernungen

Der am Anfang des Kapitels erwähnte WLAN-Streckenweltrekord in Südamerika kam mit einfachen Mitteln zustande. Das Team benutzte SoHO-Hardware und modifizierte TV-Satellitenschüsseln, um in den Anden 382 Kilometer zu überbrücken. Zum Einsatz kamen lediglich zwei Antennen, Kabel und die in in diesem Buch ausführlich vorgestellten WRT54GL-Router von Linksys als Accesspoints. Zusätzliche Verstärker wurden nicht verwendet - die Sende- und Empfangsteile der Router waren unverändert und die Sendeleistung der Router betrug nur 0,1 Watt.

Lediglich zum initialen Justieren der stark bündelnden Richtantennen wurde ein Sender von 6 Watt eingesetzt, um die Gegenseite leichter anpeilen zu können. Die Voraussetzungen für den Weltrekord waren eine freie Sichtverbindung zwischen den Antennen und der hohe Antennengewinn der beiden zweckentfremdeten Satellitenschüsseln mit einem Durchmesser von jeweils mehr als zwei Metern.

7.6.1 Link Budget - wie weit kann man funken?

Ob ein Link noch funktioniert und welche Geschwindigkeit/Datendurchsatz erreicht wird, kann durch die Berechnung des Link Budgets festgestellt werden, vorausgesetzt, es gibt verlässliche Werte als Grundlage der Berechnung. In das Link Budget fließen alle signifikanten Werte für die Funkstrecke in einer Gewinn- und Verlustrechnung zusammen.

Das fängt mit dem Gewinn der Antennen, der Ausgangsleistung der WLAN-Karten und der Empfängerempfindlichkeit auf der Habenseite an, führt

[2] Bauanleitungen gibt es zum Beispiel unter `http://www.saunalahti.fi/elepal/antenna2.html` und `http://martybugs.net/wireless/biquad/`

über die Verluste in Kabeln und Steckern bis zur Dämpfung im Freiraum zwischen den Antennen auf der Seite der Verluste. Soll der Link auch unter widrigsten Wetterbedingungen noch funktionieren, muss zusätzliche Sicherheit einkalkuliert werden – ein sogenannter Safety Margin. Bei dickem Nebel oder Schneeregen soll die Datenverbindung immer noch genügend Bandbreite erzielen. Sollte sich allerdings ein Vogel in der Antenne ein Nest bauen, kann man den Taschenrechner getrost an die Wand werfen...

In dicht bebauten Gebieten sollte die Messung des vorhandenen Rauschpegels beim Empfang in die Berechnung einfließen. Große Reichweiten lassen sich am besten mit scharf bündelnden Richtantennen erzielen, da sie neben dem Gewinn beim Empfang und beim Senden vor allem den Störpegel von anderen Anwendungen stark reduzieren.

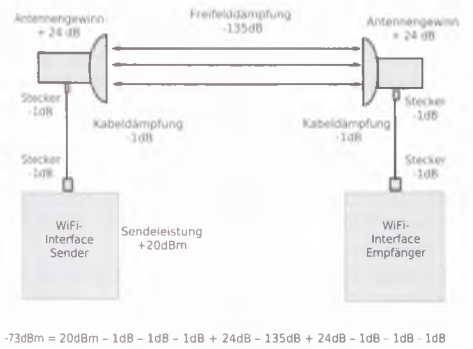


Abbildung 7.5: Link-Budget-Abschätzung

Aus der Grafik wird deutlich, dass die sendende Station lediglich mit rund 17 dBm (50 Milliwatt) arbeitet, da die Verluste in Kabel und Steckverbindungen die am Eingang der Sendeantenne eintreffende Sendeleistung reduzieren. Eine Dämpfung von 1 dB pro Steckverbindung ist allerdings eher vorsichtig geschätzt.

Die angegebenen Werte sind lediglich exemplarisch zu verstehen. Bei gut verarbeiteten und gut montierten Steckverbindern nach N-Norm beträgt die Dämpfung pro Verbindung nur 0,2 dB. Bei schlechtem Material und unsauberer Montage können auch mehrere dB pro Steckverbindung verloren gehen. Die Dämpfung im Kabel ist ebenfalls von Typ, Qualität und Länge abhängig.

Durch den Gewinn der Sendeantenne wirkt der skizzierte Aufbau in der Hauptstrahlrichtung wie ein Sender mit 41 dBm (17 dBm Signal am Antenneneingang + 24 dBi Antennengewinn), der an einer Antenne ohne Gewinn betrieben wird. Dieser – theoretische – Wert wird als effektive isotrope

Strahlungsleistung (Effective Isotropic Radiated Power, abgekürzt EIRP) bezeichnet.

Es gibt rechtliche Einschränkungen hinsichtlich der effektiven Strahlungsleistung in der Hauptstrahlungsrichtung einer Antenne. Der Betrieb des in Abbildung 7.5 skizzierten Aufbaus wäre in der BRD nicht legal.

Erlaubte Strahlungsleistung – lohnen sich gute Antennen trotzdem?

Leider ist in der BRD die erlaubte maximale effektive Strahlungsleistung, die durch Bündelung der Strahlungsenergie erzielt wird, im 2,4 GHz-Band auf nur 100 mW (entsprechend 20 dBm) beschränkt. Im Outdoorband bei 5,8 GHz sind immerhin 1 Watt effektive Strahlungsleistung gestattet. Der Einsatz einer Antenne mit hohem Gewinn zum Aufbau einer Richtfunkstrecke lohnt sich dennoch immer, auch wenn man die Sendeleistung auf sehr geringe Werte reduzieren muss, falls es die verwendeten Geräte erlauben.

Antennen mit gutem Gewinn haben in dreierlei Hinsicht einen positiven Effekt. Gerichtete Antennen mit sehr hohem Gewinn verstärken das Signal sowohl beim Senden als auch beim Empfangen. Gegen das Verstärken von Signalen beim Empfangen gibt es zum Glück keine Gesetze. Ein weiteres entscheidendes Plus einer stark richtenden Antenne ist, dass sie unerwünschte störende Signale, die nicht in der Hauptstrahlrichtung liegen, stark abschwächt. Damit lässt sich der Rauschpegel am Empfangsort oft wirksam absenken.

Eine gute Antenne bringt also mehr Leistung beim Senden und Empfangen und unterdrückt Signale, welche nicht in der gewünschten Empfangsrichtung liegen. Die Nachrichtentechniker sagen deshalb zu Recht: Eine gute Antenne ist der beste Hochfrequenzverstärker.

Keinesfalls sollte die überhöhte Sendeleistung durch Verluste in einem verlängerten Antennenkabel kompensiert werden, um durch die Kabeldämpfung eine eigentlich illegal hohe Sendeleistung künstlich zu drücken. Die Dämpfung des Kabels reduziert sowohl die Sende- wie auch die Empfangsleistung. In diesem Fall kann man das Ganze getrost vergessen, sich den ganzen Aufwand schenken und zur mitgelieferten Stummelantenne greifen. Einziger Vorteil gegenüber dem Stummel: Der Rauschpegel wird durch die Richtwirkung abgeschwächt, wenn eine Richtantenne verwendet wird. Das kann ein großer Vorteil sein, dürfte aber vermutlich die hohen Kosten nicht aufwiegen.

Empfindlichkeit von WLAN-Karten

Ein entscheidendes Qualitätsmerkmal einer WLAN-Karte oder eines Accesspoints ist die Empfindlichkeit des Empfängers. Auch die Empfangsempfindlichkeit wird üblicherweise in dBm angegeben. Hier sind es natür-

lich negative Werte. Je größer der negative dBm-Wert, desto empfindlicher ist die WiFi-Karte. Hier die Werte einer ausgezeichneten Karte (Senao NET-EL-NMP-8601-PLUS) mit Atheros-Chipsatz:

Modus	Datenrate	Empfindlichkeit
802.11b	1 Mbps	-96 dBm
	11 Mbps	-92 dBm
802.11g	6 Mbps	-92 dBm
	54 Mbps	-76 dBm

Tabelle 7.1: Empfindlichkeit der Karte Senao NET-EL-NMP-8601-PLUS

Das sind allerdings Spitzenwerte, die Empfindlichkeit der meisten WiFi-Karten ist bedeutend schlechter. Interessant ist an den Werten, dass die Karte bei gleicher Signalstärke im 802.11b Modus fast doppelt so schnell ist - 11 Megabit bei -92 dBm statt 6 Megabit. Der 802.11b-Modus ist bei langen Funkstrecken also nicht unbedingt Schnee von gestern.

Hier die Daten einer Netgear MA401 (PCMCIA) zum Vergleich:

Modus	Datenrate	Empfindlichkeit
802.11b	1 Mbps	-92 dBm
	2 Mbps	-88 dBm
	5.5 Mbps	-87 dBm
	11 Mbps	-84 dBm

Tabelle 7.2: Empfindlichkeit der Karte Netgear MA401

Die Netgear-Karte schafft bei der gleichen Signalstärke, bei der die Senao 11 MBit pro Sekunde liefert, gerade mal 1 MBit/s. Wenn die Senao-Karte die Datenrate auf 5,5 MBit/s reduzieren muss, ist bei der Netgear-Karte die Funkverbindung bereits zusammengebrochen.

Dabei ist die Netgear-Karte eher gutes Mittelfeld. Wirklich schlechte Karten liegen noch einige dB darunter. Man muss der Netgear-Karte allerdings zugute halten, dass der in ihr verbaute Prism-Chipsatz ein ganzes Stück älter ist als der Atheros-Chipsatz in der Senao.

7.6.2 Signal-Rauschabstand

Die erzielbare Übertragungsrate hängt vom Verhältnis zwischen Nutzsignal und unerwünschten Störsignalen ab – dem Signal Noise Ratio, abgekürzt SNR. Nur wenn das Nutzsignal um einen bestimmten Betrag über dem Rauschen liegt, ist eine Datenübertragung möglich. In ländlichen Gegen-

den treten kaum Störsignale auf und das Rauschen (Noise) ist geringer als das Empfängerrauschen in der WLAN-Karten (auch Empfänger rauschen). Liegt aber in einer Stadt an der Antenne ein Rauschpegel von -88 dBm an, kann ein Nutzsignal von -92 dBm natürlich nicht mehr empfangen werden.

7.6.3 Fresnel-Zone, Multi-Path-Propagation und Phasenrauschen

Die Hauptschwierigkeit beim Aufbau von Langstreckenlinks ist es, eine ausreichend breite freie Sichtlinie zwischen den beiden Endpunkten zu erzielen. WLAN-Wellen können Hindernisse wie Bäume, Gebäude oder Hügel nicht durchdringen. Eine Sichtlinie vom Durchmesser eines Bleistifts reicht aber nicht aus. Um die Sichtlinie muss ein dreidimensionaler Bereich mit der Form eines Rotationsellipsoiden (geformt wie ein amerikanischer Football) frei von Hindernissen sein – die sogenannte Fresnel-Zone (Fresnel wird übrigends Fre-nell ausgesprochen).

Abbildung 7.6: Fresnel-Zone

Befinden sich Objekte in der Fresnel-Zone, erzeugen diese durch Brechung neue Wellenfronten oder spiegeln die eigentliche Wellenfront. Das führt dazu, dass an der Antenne des Empfängers Wellenfronten gleichzeitig eintreffen, die in ihrer Phasenlage gegeneinander verschoben sind, da sie unterschiedlich lange Wege zurückgelegt haben.

Dieses Phänomen wird als Multi-Path-Propagation bezeichnet. Wellenfronten, die phasengleich eintreffen, schaden nicht, sie würden die Stärke des Empfangssignals sogar verstärken. Wellenfronten mit unterschiedlicher Phasenlage können sich dagegen gegenseitig auslöschen. Treffen zwei gleich starke Wellenfronten mit einer um 180 Grad verdrehten Phasenlage gleichzeitig an der Empfangsantenne ein, löschen sie sich gegenseitig aus.

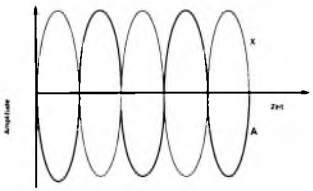


Abbildung 7.7: Phasenverschobene Wellen

Bei einer Wellenlänge von 12 Zentimetern genügt es, wenn eine Wellenfront einen Weg zurücklegt, der nur 6 Zentimeter länger ist als die direkte Sichtlinie (oder jeden Weg, der ein Vielfaches von 12 Zentimeter plus 6 Zentimeter länger ist), um genau um 180 Grad phasenverschoben an der Empfangsantenne einzutreffen.

In der Praxis erreicht bei gestörter Fresnel-Zone stets ein Gemisch von Wellenfronten mit unterschiedlichster Phasenlage die Empfangsantenne. Dieses Gemisch von Wellenfronten unterschiedlicher Phasenlage erzeugt einen Rauschteppich, das Phasenrauschen.

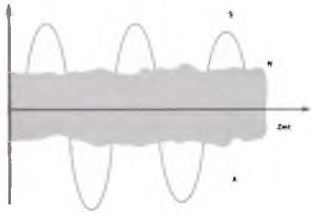


Abbildung 7.8: Phasenrauschen

Hebt sich das Nutzsignal, welches auf direktem Wege entlang der Sichtlinie gereist ist, nicht genügend von dem durch Multi-Path-Propagation erzeugten Phasenrauschen ab, funktioniert die Funkstrecke nicht mehr. Ein Erhöhen der Sendeleistung hilft in diesem Fall wenig, da es die phasenverschobenen Störsignale mit verstärkt. Idealerweise ist die Fresnel-Zone bis zu einem Radius von 80 Prozent frei von störenden Objekten, dann tritt kein signifikanter Signalverlust durch Phasenrauschen auf. Mindestens aber sollte der Radius der Fresnel-Zone zu 60 Prozent frei sein. In diesem Fall erzeugt das Phasenrauschen bereits einen deutlichen Signalverlust, abhängig von der Oberflächenbeschaffenheit. Flacher Erdboden in der Fresnelzone erzeugt einen Signalverlust von rund 2 dB, kleinere Häuser und Wald etwa 3 dB, städtische Bebauung mit hohen Gebäuden etwa 5 dB.

Der Radius r der Fresnel-Zone ist von der Wellenlänge λ und der Entfernung zwischen den Antennen abhängig und kann nach folgender Formel berechnet werden (alle Berechnungsgrößen in der Einheit Meter):

$$r = 0.5 * \sqrt{(\lambda * d)}$$

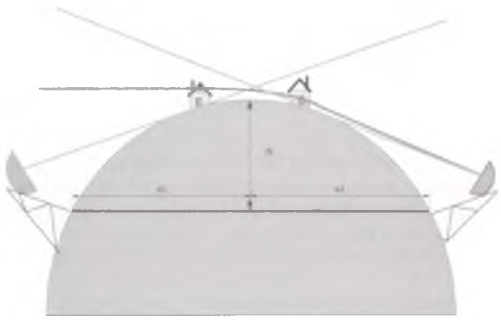
Da die Wellenlänge ein wichtiger Faktor für den Radius der Fresnel-Zone ist, liegt es nahe, die höchstmögliche Frequenz zu wählen, um die Fresnel-Zone so klein wie möglich zu halten. 802.11a ist dafür am besten geeignet.

7.6.4 Erdkrümmung

Ein Langstreckenlink ist vor allem wegen der Erdkrümmung schwer zu realisieren. Berechnet man eine Funkstrecke, stellt man fest, dass man bald an der Erdkrümmung scheitert, da die Erde unweigerlich in die Fresnel-Zone hineinragt. Der Weltrekord über 382 Kilometer war nur möglich, weil in den Anden zwei geeignete Standorte mit jeweils etwa 4000 Metern Höhe gefunden werden konnten, zwischen denen die mächtige Fresnel-Zone ausreichend Platz fand.

Glücklicherweise kommt die Physik den Weitfunkern durch die Brechung der Radiowellen etwas entgegen. Jeder kennt sicherlich den Effekt, dass ein Löffel in einem Wasserglas wegen der Brechung des Lichts beim Übergang in das Wasser gekrümmt erscheint. Auch bei Radiowellen tritt Brechung beim Übergang zwischen Medien unterschiedlicher Dichte auf.

Abbildung 7.9: Wirkung der Erdkrümmung



Da sich die Erde krümmt, erreichen die elektromagnetischen Wellen bei ihrer Ausbreitung höhere Luftschichten mit einer geringeren Dichte, da der Luftdruck mit der Höhe abfällt. Die daraus resultierende Brechung krümmt den Pfad der Wellenausbreitung zur Erde hin. Die optische Sichtlinie ver-

längert sich dadurch um 33,3 Prozent gegenüber der geodätischen (geometrischen) Sichtlinie.

Mit den beiden nachfolgenden Formeln lässt sich die Höhe der Erdkrümmung für jeden Punkt zwischen den beiden Endpunkten berechnen:

Geodätische Erdkrümmung:

$$h = 0.07843 * (d1 * d2)$$

Erdkrümmung unter Berücksichtigung des Brechungsindex:

$$h = 0.07843 * ((d1 * d2)/1.333)$$

7.6.5 Timing-Probleme bei Langstreckenverbindungen

Das Wort Lichtgeschwindigkeit klingt zunächst überzeugend schnell. Wenn man aber schnelle Datenverbindungen mit WiFi über Distanzen von einigen Kilometern realisieren möchte, kommt einem die Lichtgeschwindigkeit bald schon eher langsam vor. Der Grund ist der Halbduplex-Betrieb von WiFi: WiFi-Karten müssen immer wieder zeitraubend auf eine Antwort der Gegenstelle warten, was Bandbreite kostet. Bei einer Funkstrecke von 300 Metern beträgt die Laufzeit der elektromagnetischen Wellen schon 1 Mikrosekunde, 10 Mikrosekunden Laufzeit verstreichen bereits auf einer Distanz von 3000 Metern.

IEEE 802.11abg wurde nicht für Funkstrecken über große Distanzen entwickelt. Deshalb bekommt man bei langen Funkstrecken Schwierigkeiten mit dem Protokoll.

802.11 sieht ein *Short Inter-Frame Spacing Interval* von 10 Mikrosekunden vor. Das heißt, zwischen zwei Datenfragmenten wartet der Sender maximal 10 Mikrosekunden auf die Bestätigung von der Gegenstelle, ob das letzte gesendete Datenfragment korrekt empfangen wurde. Trifft das Acknowledgement für das Datenfragment nicht rechtzeitig ein, wird das Datenfragment nochmals verschickt. Das Protokoll nimmt an, dass entweder das Datenpaket oder das Acknowledgement von der Gegenstelle wegen einem Paketverlust oder einer Kollision im Medium Funk verloren gegangen ist, obwohl es sich in Wirklichkeit noch in der Luft zwischen den Antennen befindet.

Da Signale hin- und hergeschickt werden müssen, ist die maximale Entfernung schon allein wegen der Laufzeit der Signale geringer als 1500 Meter, wenn man bedenkt, dass die WLAN-Karten nicht unendlich schnell sind und selbst Zeit für die Bearbeitung brauchen. Das Short Inter-Frame Spacing Interval ist eigentlich bereits für eine Distanz von einem Kilometer zu kurz. Die Übertragungsrate geht zwar zurück, die Verbindung funktioniert meistens trotzdem, da irgendwann doch die Bestätigung eintrudelt – für ein Datenfragment, das bereits mehrmals unnützerweise vom Sender wiederholt wurde.

Die maximale Entfernung ist je nach verwendeter Karte unterschiedlich. Manche Karten funktionieren nach fünf Kilometern nicht mehr, andere schaffen auch 25 Kilometer ohne Tricks. Zu letzteren gehören die Karten mit Prism-Chipsätzen für 802.11b. Bei Atheros-Chips kann man unter Linux mit dem Madwifi-Treiber und den Befehl `athctrl` das Timing justieren. Auch beim Broadcom-Chipsatz ist das mit OpenWRT und Freifunk-Firmware möglich. Dazu gibt es sogar ein Eingabefenster im Webinterface der Freifunk-Firmware.

7.6.6 Planung von Wireless Long Shots

Bei der Planung eines langen Funkstrecke leistet die kostenlose Software *Radio Mobile* hervorragende Dienste[3]. Radio Mobile wird mit kostenlosen geografischen Daten gefüttert und berechnet automatisch die wichtigsten Parameter wie Fresnel-Zone, Freifeld-Dämpfung usw. Radio Mobile ist leider nur für Windows erhältlich, es arbeitet aber auch unter Linux im Windows-Emulator *Wine*.

Auch Google Earth kann bei der Planung einer Funkstrecke sehr hilfreich sein. Am besten lässt man sich bei Google Earth eine Verbindungslinie zwischen den beiden Endpunkten der Funkstrecke anzeigen und betrachtet die Linie aus einem extrem flachen Winkel von der Seite, um ein Gefühl für die Erhebungen in der Sichtlinie zu bekommen. Sowohl Radio Mobile als auch Google Earth arbeiten natürlich nicht ohne Einschränkungen, denn sie können die Höhen von Gebäuden oder hohen Bäumen nicht berücksichtigen, die möglicherweise mitten in der Fresnel-Zone liegen.

7.7 Sind elektromagnetische Wellen von WiFi-Geräten gefährlich?

Eine Abschätzung, ob die von WiFi ausgesendeten elektromagnetischen Wellen einen negativen Einfluss auf die Gesundheit haben, ist schwierig. Unumstritten ist die thermische Wirkung der elektromagnetischen Wellen, von der z. B. Mikrowellenherde Gebrauch machen. Bei in Deutschland zugelassenen WiFi-Geräten ist die maximale Ausgangsleistung auf 100 mW begrenzt – die thermische Wirkung ist daher vernachlässigbar klein, selbst wenn man sich ein WiFi-Gerät im Dauerbetrieb direkt an den Kopf hält.

Kommen jedoch große Sendeleistungen (mehrere Watt) und stark bündelnde Antennen (Parabolschüsseln) zum Einsatz, sollte man sich allerdings gehörig vorsehen. Auf keinen Fall sollte man sich im Fokusbereich einer stark

[3] `http://www.cplus.org/rmw/english1.html`

bündelnden Antenne aufhalten. In diesem Fall kann man sich Verbrennungen zuziehen!

Seit einigen Jahren wird immer wieder die Vermutung geäußert, dass gepulste elektromagnetische Strahlen auch nicht-thermische Wirkung auf den menschlichen Organismus haben könnten – sie sollen unter anderem den Hormonhaushalt und/oder das Bewusstsein stören und beispielsweise für Schlafstörungen verantwortlich sein. Diese sogenannten athermischen Wirkungen konnten bislang nicht empirisch nachgewiesen werden. Würden sie tatsächlich existieren, wäre dies im Fall von WiFi wohl kaum dramatisch, denn die Energiemenge ist bei WiFi im Vergleich zu anderen Funkanwendungen sehr gering.

Seit Beginn des 20. Jahrhunderts sind elektromagnetische Wellen alltäglich. Radio- und Fernsehsender existieren seit vielen Jahrzehnten und strahlen enorme Leistungen auf unterschiedlichen Frequenzbändern in dicht besiedelten Gebieten ab. Angestellte von Polizei und Feuerwehr setzen seit Jahrzehnten täglich Handfunkgeräte mit einer Sendeleistung von bis zu fünf Watt ein, bei denen der menschliche Körper als Teil der Antenne dient. Die leistungsstärksten Langwellensender der Welt haben eine Strahlungsleistung von bis zu 2 Megawatt, ein Fernsehsender bringt es auf 10 000 oder 100 000 Watt. Kaum jemand fürchtet sich jedoch vor „Funkgerätestrahlen", „Fernsehstrahlen", „Radiostrahlen" oder den Aussendungen, die aus dem Mikrowellenherd entweichen.

Mikrowellenherde verwenden das gleiche Frequenzband wie 802.11b und g, da es sich um ein ISM-Band handelt (nicht etwa weil 2,4 GHz die Resonanzfrequenz von Wasser wäre, wie vielfach fälschlicherweise angenommen wird). Ein Mikrowellenherd sendet intern mit einer Leistung bis zu 1000 Watt und darf auch bis zu 1 Watt nach außen emittieren. Ein alter Mikrowellenherd mit defekter Abschirmung „leckt" sicherlich noch einiges mehr.

Menschen, die sich vor Mobiltelefonen, DECT-Basisstationen und WLAN-Geräten fürchten, verweisen darauf, dass die seit langem in Betrieb befindlichen Sendeanlagen für Radio und Fernsehen im Gegensatz zu den modernen Anwendungen nicht „gepulst", also digital moduliert sind. Warum das bei der Wirkung auf den menschlichen Organismus einen Unterschied machen soll, ist schwer verständlich.

Die meisten Menschen sind sich gar nicht bewusst, wie viele starke elektromagnetische Felder im gesamten Spektrum permanent durch die Gegend schwirren, da sie dafür keine Sinne haben. Weil sie sich dessen nicht bewusst sind, haben sie auch keine Veranlassung, nervöse Verstimmungen auf die Anwesenheit einer bestimmten elektromagnetischen Anwendung zu schieben. Steht dagegen ein WLAN-Gerät mit erkennbaren Antennen auf dem Tisch, taucht plötzlich eine Erklärung für diese oder jene Befindlichkeit auf.

Doch wer ein Mobiltelefon besitzt, hat keinen Grund, sich über die „athermischen Wirkungen" eines WiFi-Routers in der benachbarten Wohnung oder auf dem Hausdach Sorgen zu machen. Ein Mobiltelefon hat eine Ausgangsleistung von bis zu 2 Watt (GSM 900) beziehungsweise 1 Watt (GSM 1800). Die maximale Sendeleistung eines Mobiltelefons ist bis zu zwanzigmal höher als die Sendeleistung der leistungsstärksten WiFi-Geräte auf dem Markt. Es sendet immer wieder Daten, um dem nächsten Mobiltelefonmast mitzuteilen, dass es erreichbar ist, und wird von den meisten Anwendern täglich viele Stunden am Körper getragen. Beim Telefonieren hält man sich das Gerät unmittelbar an den Kopf und schirmt dabei oft unabsichtlich die Sendeantenne mit der Hand ab, wodurch das Mobiltelefon die Sendeleistung auf das Maximum hochregelt.

Die Strahlungsdichte eines elektromagnetischen Senders nimmt mit dem Quadrat zur Entfernung ab. Ein WiFi-Router, der auf dem Fensterbrett oder auf dem Hausdach vor sich hin funkt, erzeugt dort, wo Menschen sich die meiste Zeit aufhalten, nur ein sehr schwaches elektromagnetisches Feld. Wer sich davor fürchtet, könnte ebenso gut annehmen, dass man von einer Taschenlampe, die von einem Laternenmast herunterscheint, einen Sonnenbrand bekommt. Es ist auch nicht so, dass die WLAN-Wellen ungehindert in den Körper eindringen würden. Ein großer Teil wird von der Oberfläche des menschlichen Körpers reflektiert.

Bei einer im Notebook eingesteckten oder eingebauten WLAN-Karte ist man bei stundenlanger Arbeit mit dem Gerät natürlich sehr nahe an der Sendeantenne und deshalb einer höheren elektromagnetischen Feldstärke ausgesetzt. Wer sicherheitshalber den Einfluss der elektromagnetischen Wellen so klein wie möglich halten möchte, sollte sich mit dem Notebook über ein Netzwerkkabel mit dem entfernt stehenden Router verbinden oder einen USB-WLAN-Adapter an einem USB-Verlängerungskabel verwenden.

Kapitel 8

Die Community

Der Name Freifunk hat sich im deutschsprachigen Raum als Oberbegriff für offene und freie Community-Netzwerke eingebürgert. Grundlage der Freifunk-Idee ist das *Pico-Peering-Agreement*[1]. Unter Peering versteht man ein Abkommen unter Netzbetreibern, die sich gegenseitig Datentransport durch das eigene Netz zusichern, um die Reichweite des eigenen Angebots zu vergrößern. Peering-Vereinbarungen sind daher als Grundlage des Datenverkehrs im Internet unter Netzbetreibern üblich.

Das Pico-Peering-Agreement gibt in Art einer Konvention die Grundlage für Community-Netzwerke vor. Wer daran teilnimmt, bekundet die Absicht, den Datenverkehr der anderen Teilnehmer weiterzuleiten, ohne diesen zu kontrollieren, zu filtern oder zu verändern. Durch die Vereinbarung entsteht ein freies, von Privatpersonen aufgebautes Netzwerk als digitales Allgemeingut: kostenloser, freier und ungehinderter Datenverkehr. Viele Freifunker empfinden das als wichtige Grundlage einer sich im positiven Sinne entwickelnden demokratischen Gesellschaft.

[1] `http://www.picopeer.net/PPA-de.html`

In vielen Städten gibt es heute ein Freifunk-Netz. Vorreiter und Vorbild ist für viele Communities das Berliner *Olsrexperiment*[2] – eine „Wolke" drahtlos vermaschter Netzwerkknoten, die seit 2004 durch ein Meshroutingprotokoll dynamisch geroutet werden. In der Anfangszeit experimentierten die Freifunker mit dem Protokoll Mobilemesh. Dieses wurde nach einigen Monaten durch OLSR ersetzt. Mittlerweile ist das neue Protokoll B.A.T.M.A.N. parallel zu OLSR im Einsatz. Vermutlich wird B.A.T.M.A.N. OLSR im Laufe der Zeit ablösen – um eines Tages einem neuen Protokoll Platz zu machen.

Die Berliner Meshwolke war mit rund 400 Knotenpunkten lange Zeit das größte zusammenhängende Meshnetzwerk, das von einer Community betrieben wurde. Möglicherweise ist es immer noch das größte funktionierende Meshnetzwerk, sicher ist das allerdings nicht, denn an vielen Orten bilden sich neue Community-Netze, Das Leipziger Freifunknetz vermeldet inzwischen eine Gesamtzahl von 600 Routen (die Anzahl der Routen ist größer als die der Knotenpunkte, da manche Router mehrere Interfaces haben und deshalb mehrfach in der Routingtabelle auftauchen).

Über die Anzahl der Nutzer gibt es nur Schätzungen. Lediglich die aktiv am Routing teilnehmenden Netzwerkknoten sind für andere Teilnehmer sichtbar. Wie viele Menschen täglich in Berlin die Meshknoten als Gateways benutzen, um einfach nur ins Internet zu gehen oder innerhalb des WMAN (*Wireless Metropolitan Area Network*) miteinander zu kommunizieren, ist unbekannt. Es sind vermutlich 1000 bis 5000 Nutzer, die sich täglich etwa 25 Internetzugänge teilen. Oft versorgt ein einzelner Freifunk-Router auf dem Dach ein ganzes Haus ohne Breitbandanschluss. Viele Menschen mit geringem Einkommen nutzen Freifunk, um kostenlos ins Internet zu kommen oder sich die Kosten für Internetanschlüsse zu teilen.

Wer im deutschsprachigen Raum wohnt und wissen möchte, ob es bereits ein Freifunknetzwerk am Wohnort oder in der Nähe gibt, kann das auf der Webseite der Freifunker[3] überprüfen. Hier kann und sollte man ein persönliches Freifunk-Ticket anlegen oder die Freifunk-Tickets von anderen Freifunkbegeisterten in der Umgebung suchen. Durch ein solches Ticket können sich die Freifunker in ihrer Umgebung gegenseitig bekannt machen.

Selbst wenn es noch kein bestehendes freies Funknetz geben sollte, können sich Interessierte dadurch zusammenfinden und die ersten Schritte unternehmen. Irgendjemand ist immer die erste Person, die einen Freifunkknoten einrichtet und auf die Verbreitung der Idee hofft. Jedes größere Meshnetzwerk hat schließlich einmal mit einem oder zwei Knotenpunkten angefangen.

Das Betreiben eines offenen Freifunk-Knotens lässt sich übrigens auch zur zielgruppenorientierten drahtlosen Werbung nutzen, wenn der verwende-

[2] `http://www.olsrexperiment.de`
[3] `http://www.freifunk.net`

te Netzwerkname eine eindeutige URL ist. Im Freifunk.net-Wiki kann einfach eine neue Seite (etwa: `musterort.freifunk.net`) eingerichtet werden, deren Name als Netzwerkname bei der WLAN-Konfiguration verwendet werden kann.

Die größte Verbreitung hat Freifunk in den Gebieten, die bis heute von Breitband-Internetzugängen abgeschnitten sind. Aber auch in Gebieten, in denen die Penetration mit Breitbandanschlüssen recht gut ist, gibt es engagierte Freifunker, die aus unterschiedlichen Motiven den Auf- und Ausbau eines solchen Netzwerks vorantreiben.

Viele Freifunk-Initiativen treffen sich regelmäßig, um ihre Aktivitäten zu koordinieren. Etablierte Freifunk-Communities halten zumeist auch Einführungskurse ab und helfen Einsteigern gerne mit Rat und technischer Unterstützung. Die Communities haben ein selbstverständliches Interesse am Wachstum und Ausbau ihres Netzes, da sich damit die Abdeckung und Qualität der Versorgung erhöht. Jeder zusätzliche Knotenpunkt verbessert die drahtlose Infrastruktur – auch wenn er nur auf einer Fensterbank steht, wo er das Freifunksignal über den Dächern empfängt und in eine Straßenschlucht weiterreicht.

Nicht jede Person, die Freifunk benutzen will, braucht einen eigenen WLAN-Router. Innerhalb eines Hauses kann das Netz auch per Ethernetkabel verbreitet werden. Idealerweise befestigt man den Router möglichst hoch über dem Dach und speist ihn mit ungefährlicher Gleichspannung über das Netzwerkkabel (siehe Abschnitt 5.5 ab Seite 116), welches auch den Netzzugang für das Haus bereitstellt.

Viele Freifunker begeistert auch die zwischenmenschliche Komponente, der sogenannte *Layer 8* im Freifunker-Jargon. Für die einen ist Freifunk also ein spannendes Hobby, für die anderen ein Weg, die Versorgung mit der dringend benötigten Internet-Infrastruktur selbst in die Hand zu nehmen.

Index

2,4-GHz-Band 88
  Belegung 88
  Freigabe 14
802.11-Linux-Stack 102
802.11 Draft-n 88
802.11 Pre-n 88
802.11a 90–92
  erlaubte Sendeleist. 90,91
  Frequenzbereich 90
  Kanäle 91–92
  und Radar 90
802.11b 88–90
802.11b/g 88–90
  Übertragunsgeschwindigkeit 89
  Bruttodatenraten 89
  Frequenzbereich 88
  Kanäle 88
  Kanalauswahl 89
  Kanalraster 89
  Sendeleistung 88
  Unterschied b und g 89
802.11g 88–90
802.11h 90
802.11n 92
802.11 MIMO *siehe* 802.11n

A

Abschattung 177
Accounting 155
Ad-Hoc-Modus 19
  fehlerhafter 70
  und Routing 21
  Vor- und Nachteile 20
Adhoc On-Demand Distance Vector *siehe* AODV
Aironet *siehe* Cisco-Chipsätze
Annan, Kofi 14
Antennen
  rundstrahlende *siehe* Omni
Antennengewinn
  Herstellerangaben 178
AODV 30
Apple-Notebooks
  Atheros-Karten 107
apt-get 156
AR50XX *siehe* Atheros-Chipsätze
Asus WL500G Premium 130
  Eingangsspannung 130
  TFTP-Server aktivieren 134
at76cXX *siehe* Atmel
atftp 135
athctrl 188
athermische Wirkung 189
Atheros-Chipsätze 106–108
  Cell-ID setzen 107
  CPU-Last 107
  im Macintosh 107
  Reichweite 108
  Treiber patchen 107
  Windows-Probleme 107
Atmel-Chipsatz 108
AVM-Fritz-Box 111

B

B.A.T.M.A.N.
  Connect-Modus 82
  Debugging-Ausgabe 82
  Debugging-Modus 81
  Gateway-Konfiguration 83, 86
  Originator-Intervall 80
  Ports 86
  Route-Konfiguration 83
  Routingklassen 83
  und Firewall 86
  Visualisierungsserver 86
Bäume
  Dämpfung durch - 177
Bandbreite begrenzen 156
Basisrate 146
Beacon-Intervall 147
Bel 170
Bereitschaftswert 141
Berlin als Freifunk-Pionier 192
Beugung 177
Bidirectional Link Check 74–76
Biquad-Antenne 180
Bittorrent 155
Blacklist 154
boot_wait 163
Bootloader
  seeCFE 162
Brechung 186
Broadcastadresse, ungültige 69
Broadcom-Chipsatz 108–109
  freier Treiber 109
Bruttodatenrate 92
BSSID 144
Buffalo WHR-G54S 129–130
  Eingangspannung 129
  Firmware-Image 130
  TFTP-Server 134

C
Cab-Archiv unter Linux 103
Cardbus 94
Cell-ID *siehe* IBSSID
Cell-Splitting 69, 112–115
  Diagnose 113
  Folgen 114
  verhindern 115–116
CFE 162–163
  flashen 168
  MAC-Adresse 168
  TFTP-Server 163
Chipsätze
  unterstützte 95
Cisco-Chipsätze 109
Clear-to-Send *siehe* RTS/CTS
Click Modular Router 33
Community-Netze 191–193
Compressed RAM Filesystem *siehe* cramfs
cramfs 159
CTS *siehe* RTS/CTS
CuWIN 32
CVS 45

D
d80211 *siehe* mac80211
dBd (Antennengewinn) 178
dBi (Antennengewinn) 178
dBm 170
Default-Gateway
  bei Olsrd 61
Destination-Sequenced Distance-Vector *siehe* DSDV
Devicescape 102
Dezibel 170
  Definition 170
  Rechenbeispiele 170
DHCP
  Benutzeranzahl 149
  Lease-Dauer 149
  Meshrouter als Server 140
  Start-IP 149
Dielektrikum
  im Koaxialkabel 98
Digital Divide 14
Dijkstra, Edsger W. 29
Dijkstra-Algorithmus *siehe* Link-State-Algorithmus
Dijkstra-Optimierung 143
Dipol 174
  Antennengewinn 175, 178
Distanz-Vektor-Verfahren 29
DMZ-Umleitung 139
DNS-Server 138
Dosenantenne 180
DotDraw (Olsrd-Plugin) 62
DSDV 29
DSR 30
DTIM-Intervall 147
DYM 30
Dynamic MANET On-demand *siehe* DYM
Dynamic Source Routing *siehe* DSR
DynGW 142, 153

E
E-Mule 155
Elektromagnetische Wellen 172
Elektromigration 100
Empfindlichkeit
  von WLAN-Karten 182
Energieorientierte Protokolle *siehe* Routingprotokolle
ESSID 112, 144
et0macaddr (SE505-NVRAM) 153
/etc/hosts 65
/etc/resolv.conf 65
ETX 40
  manipulieren 141
Expected Transmission Count *siehe* ETX
Exponential Backoff 24
Exposed Node 26
Extended Service Set ID *siehe* ESSID

F
Failsafe 162
Failsafe-Modus 152
Feuchtigkeit in HF-Steckern und -Kabeln 99
fttrace 61
Filesharing 155
Firewall 69, 149
Firmware
  Kernel löschen 167
Firmware-Installation
  Abschlussmeldung 132
Fischaugen-Routing 143
Fisheye-Algorithmus 42–44
Fragmentierung 25
Fragmentierungs-Schwelle 147
Frame-Burst 146
Freifelddämpfung 176
Freifunk-Firmware 125
  Editor vi 156
  Hardwareanforderungen 127
  Installation 130–135
  IP-Bereich 148
  kompatible Hardware 126
  LAN-IP 148
  Login per SSH 136
  NAT 149
  Neustart 152
  Prüfsummen 131
  TFTP-Installation 134
  Update 152
  Webinterface 135–153
  Webserver 151
  Zusatzsoftware 151–152
freifunk-gateway-de 154
Freifunk-Gateway-Plugin 154
Freifunk-Initiativen 191–193
freifunk-recommended (Software) 151
Frequenz 173
Fresnel-Zone 184
  Objekte in der ~ 185
  und Wellenlänge 186
FSF *siehe* Free Software Foundation
Full Mesh 20
Funk-Modus 145

G
Gateway 153–156
  über DSL 153

über LAN 153
Gateway-Switching 67
Gehäuse für Router *siehe* Router-Gehäuse
Geografische Multicast-Protokolle *siehe* Routing-protokolle
Geografische Protokolle *siehe* Routingprotokolle
gepulste Strahlung 189
Glasfaser 13
Google Earth 188
GPL 159
Graphviz 85

H
Halbduplex-Betrieb 187
Halbduplex-Modus 22
Bandbreite 23
Hardwareanforderungen für Freifunk-Firmware 127
Hazy Sighted Link State Routing 30
Hello-Nachrichten 39
und Originatornachrichten 73
Hertz, Heinrich Rudolf 174
HF-Kabel *siehe* Koaxialkabel
HF-Steckverbinder 97
Dämpfung 181
Reverse 97
Hidden Node 26
Hierarchische Protokolle *siehe* Rotingprotokolle
HIPERLAN 90
HNA 140
Hochfrequenz-Verbinder *siehe* HF-Steckverbinder
Hopcount 23
horst 152
HSLS 30, 32
Httpinfo (Olrsd-Plugin 142
Httpinfo (Olsrd-Plugin) 63
Einrichtung 63
Hybride Protokolle *siehe* Routingprotokolle
Hysterese 37–38
und Multipoint-Relais 38

I
IBSS Merge 113
IBSS-Splitting *siehe* Cell-Splitting
IBSSID 112–113, 144
Fehlfunktionen 113–115
fest einstellen 115
Festlegung 113
Zeitstempel manipulieren 115
IEEE 802.11a *siehe* 802.11a
IEEE 802.11b *siehe* 802.11b
IEEE 802.11b/g *siehe* 802.11b/g
IEEE 802.11g *siehe* 802.11g
IEEE 802.11n *siehe* 802.11n
il0macaddr (SE505-NVRAM) 153
Independent Basic Service Set Identifier *siehe* IBSSID
Infrastrukturmodus 19
INRIA 36
Intel-Chipsätze 109–110
Interferenzen 89
Intersil 104
IP-Blacklist 154
iPKG 156, 160
ipw2100 109
IPW2200
fehlerhafter Ad-hoc-Modus 101
ipw2200 109
ipw3945 110
Is-Direct-Link 75
ISM 88
ISM-Frequenzband 88
Isotrop-Strahler 178
iwconfig 113

J
JTAG-Kabel 164
JTAG-Schnittstelle 164

K
„Kamikaze" 95, 161
Kanalraster 89
Kazaa 155
Koaxialkabel 98
Dielektrikum 98
Impedanz 98
Verlegung 99
Kreisrouten 42, 43

L
Layer 8 155, 193
Leipziger Freifunknetz 192
libpthread 157
Link Budget 180–182
Link-Qualität 40
Link-State-Algorithmus 29, 35
Link-State-Routing 72
Grenzen 41
Linkquality-ETX-Algorithmus 37
Linksys
und GPL 159
Linksys WRT54GL *siehe* WRT54GL
Linksys-Firmware
Versionsprobleme 168
Linux-Treiber 102
freie und proprietäre 102
Location-Based-Services 30
LQ 40
Lucent-Orinoco *siehe* Orinoco-Chipsätze

M
MAC-Blacklist 154
mac80211 102
Madwifi 106–107
Many-to-Many-Topologie 20
Maximum Transfer Unit *siehe* MTU
Mcast-Forward 143
Meraki 14, 31, 33
Meshlinux 97
Meshrouter 21
als Accesspoint 140
Mikrotik 96
Mikrowellenöfen 88
Mikrowellengeräte 189
MIMO-Karte 92

tatsächliche Reichweite 92
Mini-Foo 139
Minimum Hop Count
Nachteile 37
MiniPCI 94
MIT Roofnet 31, 33
Mobilemesh 192
MTU 25
MTU-Wert 147
Multi-Path-Propagation 184
Multicast-Protokolle *siehe* Routingprotokolle
Multihop-Links 22
Multihop-Route
Bandbreitenreduzierung 23
Paketverluste 23
Multipoint-Relais 38
und Hysterese 38
Multipoint-to-Multipoint *siehe* Many-to-Many-Topologie

N
N-Buchse 98
Nachbar-Link-Qualität 40
Nameservice 142
Nameservice (Olsrd-Plugin) 65
NAT 67, 149, 157–159
ausschalten 158
Ports öffnen 157
Ndiswrapper 103–104
Neighbor Link Quality *siehe* NLQ
NetBSD 33
Netgear MA401 183
Nettodatenrate 92
Network-Address-Translation *siehe* NAT
Netzwerksimulation 31
NLQ 40, 59
NS2 33
NVRAM 162
Backup 167
et0macaddr 153
il0macaddr 153
löschen 167

O
OLSR 29, 72
Implementierungen 36
Olsr.org-Implementierung 36
OLSR Tempo 141
OLSR Traffic Shaping 143
OLSR-Daemon *siehe* Olsrd
OLSR-DHCP 140
OLSR-DMZ 158
Olsr-Filter 139
Olsr-Switch 46
OLSR-Viz 151
OLSR-Willingness 141
Olsr.org 36
Olsrd
Abstürze unter Windows 46
CVS-Version 45
Debuglevel 58
ETX 40
für Macintosh 46
für MS-Windows 46
Funktionsweise 39
Installation 44
Internet-Gateway unter Windows 47
Konfigurationsdatei 48–51
NLQ 40
Plugins 61
Plugins installieren 45
TC-Messages 41
unterstützte Systeme 39
Olsrexperiment 192
Omni 178–179
horizontale Bündelung 179
Strahlungsdiagramm 178
One-to-Many-Topologie 19
OpenWRT 159–161
als Meta-Distribution 159
Routingsoftware installieren 161
unterstützte Chipsätze 95
Optimized Link State Routing *siehe* OLSR
Optische Anschlussleitung *siehe* Glasfaser
Originatorintervall wählen 80
Originatornachricht 73
Inhalt 73
Statistik 74
Verbreitung 73
Weg einer ~ 76–79
Orinoco-Chipsatz 105–106
Karten auf Ebay 106
Linux-Unterstützung 105
und Cell-Splitting 106
OSPF 72
Overlay-Dateisystem 139

P
P2P-Traffic blocken 155
Packet Radio 14
Paketverluste 23
Parallelport 166
PC-Engines 96
PCI Express Mini Card 110
PCI-Karten 94
PCI-WLAN-Karten aufrüsten 94
PCMCIA Typ II 94
PCMCIA-Karten
16 Bit 94
Peer-to-Peer-Modus *siehe* Ad-Hoc-Modus
Peering 191
Pegel 170
Pfostenstecker 164
Pico-Peering-Agreement 191
Pigtail 94
Plugins
Olsrd 61
PoE 116–122
Point-to-Multipoint-Topologie *siehe* One-to-Many-Topologie
Polarisation 175–176
lineare 175
Verluste durch ~ 175
zirkulare 175
Port-Forwarding 157
Power over Ethernet *siehe* PoE
PPPoE nachrüsten 153
Prism-Chipsätze 104–105
Prism-Utils 104
Prism54-Chipsätze 106

Cell-ID setzen 106
Proaktive Protokolle *siehe* Routingprotokolle
PuTTY 136, 156
Pyramid-Linux 97

Q
QOS-Protokoll (ETX) 141

R
Radar 90
Radio Mobile 188
Ralink-Chipsätze 110–111
Reaktive Protokolle *siehe* Routingprotokolle
Realtek-Chipsätze 111
Reflektion 177
Request-to-Send *siehe* RTS/CTS
Reverse HF-Verbinder 97
RFC 3626 36
Richtantenne 180
RJ45 150
Round-Trip-Wahrscheinlichkeit 40
Router-Firmware
landestypische 89
Router-Gehäuse 99
empfohlene 100
Entlüftung 100
Kondenswasser 100
Routermodelle, Freifunkkompatible 126
Routerplatinen 96
Routing-Loops *siehe* Kreisrouten
Routingalgorithmen
proaktiv 71
reaktiv 71
Routingprotokoll
hybride 30
Routingprotokolle
akademische Diskussion 31
Anforderungen 27
energieorientierte 30
geografische 30
Geografische Multicast-Protokolle 30
hierarchische 30
Kategorien 28
multicast 30
proaktive 29
reaktive 29
Skalierbarkeit als Kriterium 31
tatsächlich verwendete 32
rrdtool 151
rt2400 110
rt2500 110
rt2570 110
rt2600 110
rt2800 110
rt2x00 (Treiberprojekt) 110
RTL8180L 111
RTL8185L 111
RTL8187L 111
RTS-Schwelle 147
RTS/CTS 26–27
Rundstrahler 175

S
„Schrottband" 88
SE505
MAC-Adressen 153
Sektorantenne 179
Senao 104
Senao NET-EL-NMP-8601-PLUS 183
serielle Konsole 97
Short Inter-Frame Spacing Interval 187
Shortest Path Algorithmus 35
Shortest-Path-Algorithmus *siehe* Link-State-Algorithmus
Sichtlinie
geodätische 187
optische 187
Signal Noise Ratio *siehe* Signal-Rauschabstand
Signal-Rauschabstand 89, 183
Sleeve-Dipol 174
Soekris Engineering 96
Soft-MAC
und Linux 106
SoHO-Router
Vor- und Nachteile 93–94
Sourcerouting 72
SrcRR 30, 31
SSH 136, 156
Störreichweite 28
Statische Routen 148
Strahlungsdiagramme, erfundene 178
Strahlungsleistung, erlaubte 182
Streuung 177
Super-G 90

T
Tücke,Sven-Ola 125
TC-Message 41
Intervall-Länge 42
TTL 42
TCP 24
Retransmissions 25
und Datenstau 24
Texas-Instruments-Chipsätze 111
TFTP 134–135, 163
Client 135
tftpd32 135
Thumfart, Maximilian 46
„Time to live exceeded" 42
Time-To-Live *siehe* TTL
Timing 145, 187–188
Toennesen, Andreas 36
Topologiedatenbank 85
Topologievisualisierung 62
Topology Control Message *siehe* TC-Message
Tourrilhes, Jean 105
Traffic-Shaping 155
Trivial File Transfer Protokoll *siehe* TFTP
True-Diversity 128
TTL 42
tun (Kernelmodul) 79
Tunnel-Interface 57
Tupperware als Routerbehälter 99
Turbo-Mode 90

U
Übertragungsrate 146
Unidirectional-Flag 75
URL-Whitelist 154

V
versteckter Knoten *siehe* Hidden Node
Virtual LAN *siehe* VLAN
Visualisierung von Meshnetzwerken 85
Visualisierungsserver 85
  B.A.T.M.A.N. 86
VLAN 150–151
Vollmesh *siehe* Full Mesh
vulkanisierendes Klebeband 99
VxWorks 126, 128

W
Wellenlänge 173
„White Russian" 125, 160
Whitelist 154
Wizards of OS 3 36
WL500G *siehe* Asus WL500G Premium
WLAN-Weltrekord 109
WMAN 192
WRT54G
  Unterschied zu WRT54GL 128
wrt54g 166
wrt54g (JTAG-Programm) 166–168
WRT54GL 128–129
  DMZ-LED 162
  Eingangsspannung 128
  Ethernet-Schnittstellen 131
  Firmware-Image 129
  Firmware-Installation 131–133
  Firmware-Versionen 168
  Kurzschließen 164
  Preis 16
  reparieren 164
  Sendeleistung 128
  TFTP-Server 163
WRT54GS 126
WRTG54GL
  Kernel löschen 167
  NVRAM löschen 167
  Passwort 132
  Standard-IP 131

Z
ZD1211/ZD1211B 111
zd1211rw (Treiber) 112
Zone Routing Protocol *siehe* ZRP
ZRP 30
Zydas-Chipsätze 111–112

Mesh

Drahtlose Ad-hoc-Netze

In drahtlosen Ad-hoc-Netzen bilden die Geräte selbst einen Teil der Infrastruktur. Solche Mesh-Netzwerke benötigen keinen zentralen Zugangspunkt, sondern organisieren sich selbst. Sie sind nicht etwa akademische Spielerei, sondern Realität, und jeder kann daran teilnehmen. Community-Netze decken in Großstädten wie Berlin ganze Stadtviertel ab oder ermöglichen in abgeschiedenen Regionen vielen Menschen Zugang zum Internet.

Mesh-Netze brauchen keine anspruchsvolle Technik – preiswerte WLAN-Router oder -Karten aus dem Elektronikmarkt genügen. Die benötigte Software für das Routing und die Konfiguration ist – selbstverständlich – Open Source, und eine rege Community hilft bei Problemen, im Internet und in vielen Orten auch lokal.

Dieses Buch beschreibt alles, was nötig ist, um Mesh-Netze zu verstehen, die Hardware auszuwählen, die Software zu beherrschen und darüber hinaus noch ein bisschen mehr. Es erklärt die Routing-Protokolle OLSR und B.A.T.M.A.N. und die zugehörige Software, geht auf die Eigenheiten von WLAN-Chipsätzen ein und beschreibt, wie man Antennen gekonnt zur Reichweitensteigerung einsetzt.

Doch auch wer neue Firmware auf seinen Router „flashen" will (und ihn bei Misserfolgen wieder reparieren muss) erfährt, wie das geht. Gespickt mit zahllosen Tipps für die WLAN-Praxis, eignet sich dieses Buch also auch für alle, die mehr aus ihrer Hardware machen wollen.

Elektra lebt und arbeitet in Berlin, ist in der Freifunk-Community aktiv und war unter anderem an der Entwicklung des B.A.T.M.A.N.-Routingprotokolls beteiligt. Ihr Engagement für WLAN-Technik und drahtlose Ad-hoc-Netze hat sie schon nach Indien und Bangladesh geführt.

www.opensourcepress.de

ISBN 978-3-937514-39-0

€ 19,90 [D]
€ 20,60 [A]
SFr 32,90